IFSP – INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

Franz Einriech Arnold

SISTEMA PARA SUPERVISÃO E CONTROLE DE ROBÔ BASEADO EM LINGUAGEM NATURAL.

Dissertação de Mestrado apresentada ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – IFSP como parte dos requisitos para obtenção do título de Mestre em Automação e Controle de Processos.

Orientador: Prof. Dr. Paulo Marcos de Aguiar.

SÃO PAULO – SP
2012

A775 ARNOLD, Franz Einriech.
Sistema para Supervisão e Controle de Robôs Baseado em Linguagem Natural / Franz Einriech Arnold – 2012.
270 f.: il.; 30 cm

Orientador: Prof. Dr. Paulo Marcos de Aguiar

Dissertação (Mestrado Profissional em Automação e Controle de Processos) - Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, IFSP, 2012.

1. Interface Robótica 2. Acionamento Remoto 3. Linguagem Natural I. ARNOLD, Franz Einriech. II. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo III Título

CDU 681.5

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São Paulo

ATA DE EXAME DE DISSERTAÇÃO DO MESTRADO PROFISSIONAL

Nome do Aluno: Franz Einriech Arnold

Nome do Orientador: Prof. Dr. Paulo Marcos de Aguiar

Título do Trabalho: "SISTEMA PARA SUPERVISÃO E CONTROLE DE ROBÔ BASEADO EM LINGUAGEM NATURAL"

Abaixo o resultado de cada participante da Banca Examinadora

Nome completo do Participante da Banca	Sigla da Instituição	Aprovado/Reprovado
Prof. Dr. Paulo Marcos de Aguiar Orientador	IFSP – SPO	Aprovado
Prof. Dr. Clayton Eduardo dos Santos Membro Interno	IFSP – BRA	APROVADO
Profª Dra. Roseli Aparecida F. Romero Membro Externo	ICMC – USP	APROVADO

Considerando-o [X] **APROVADO**

[ ] **NÃO APROVADO**

São Paulo, 19 de setembro de 2012

Assinaturas

Presidente da Banca

Clayton Eduardo dos Santos
Membro Interno

Roseli Ap. Francelin Romero
Membro Externo

AGRADECIMENTOS

À Deus, nosso Pai de Amor e Bondade, pela permissão sem a qual nada seria possível, pela generosa oferta de infindáveis oportunidades de aprendizado, autoconhecimento e desenvolvimento, pelo amparo nas horas difíceis e compreensão da grande imperfeição que nos caracteriza a manifestação.

Ao grande amigo Jair de Azevedo Jr., cujo apoio, incentivo, orientação e perseverança inesgotável foram de valor inestimável nas vezes quando meu ânimo ameaçou falhar.

Aos familiares e amigos pelo suporte direto e pela compreensão com o tempo ausente, os quais foram dedicados longamente à execução deste trabalho.

Aos companheiros de trabalho no IPEN, que pacientemente compreenderam meu cansaço depois das longas horas de estudo, sempre oferecendo amparo e novas oportunidades, tantas vezes quantas foram necessárias.

Ao Prof. Dr. Paulo Marcos de Aguiar, pela valiosa orientação, sem a qual este trabalho não poderia ser realizado.

Ao Prof. Dr. Paulo Barbosa, pelas orientações e colaboração nos momentos decisivos que se constituíram em importante suporte aos bons resultados.

Ao Prof. Dr. Ricardo Pires, cuja sabedoria e busca contínua pelo conhecimento representaram valioso esteio à perseverança, ao estudo e à dedicação, desde o dia da avaliação do meu projeto de pesquisa como durante as aulas de laboratório, onde colaboramos como professores.

Aos meus colegas da turma de Mestrado em Automação e Controle, cujo humor e entusiasmo foram sempre grande estímulo ao estudo e ao esforço continuados.

Aos meus alunos do IFSP, pelo carinho e compreensão que inspiraram a energia para prosseguir no árduo caminho de aprender a ensinar.

"A imaginação é mais importante que o conhecimento."

Albert Einstein

RESUMO

O acionamento de robôs móveis pode ser por controles elétrico ou eletrônico, manual ou então por programação utilizando comandos e linguagem específicos. Porém, um usuário leigo provavelmente não obteria resultados satisfatórios eficazmente. Uma alternativa, nesse caso, é comandar o robô diretamente em Português, ou seja, utilizando a *linguagem natural*.

Com a proposição e avaliação de sistema para supervisão, controle e operação de robô por meio de linguagem natural em Português, alguns recentes avanços tecnológicos nas áreas da *robótica* e da *inteligência artificial* são abordados de forma prática através de solução concebida com foco em serviços.

Foram abordados conceitos como a progressão "dado – informação – conhecimento", noções suficientes sobre representação do conhecimento, definição e estruturas da linguagem natural comparativamente às linguagens artificiais, o Processamento da Linguagem Natural ou Linguística Computacional, sua implementação através de métodos determinísticos e estocásticos (estatísticos), modelos de Entropia Máxima e a importância do uso de ontologias para obtenção dos resultados esperados.

Para a prática, foi proposta uma solução que servisse como prova de conceito funcional do uso combinado das teorias e das tecnologias em questão. Nenhum recurso especial foi usado tendo-se em vista o objeto de propor uma solução facilmente acessível, mesmo que baseada em novas tecnologias. O *software* de integração é apresentado frente à solução prática proposta. São avaliadas as *funcionalidades* que se tornaram possíveis, a *potencialidade* das novas funcionalidades que podem ser criadas, sua *adaptabilidade* às características dos robôs que venham a ser usados e as diversas *interfaces homem-máquina* que poderiam ser usadas para acionamento dos serviços existentes.

O resultado foi positivo em mais de três quartos dos casos, inclusive com as variações permitidas na linguagem natural. Comandos complexos, com encadeamento de indicação de referências externas, diretivas e navegação automática foram avaliados com diferentes estilos de redação e tempos verbais.

Palavras-chave: Interface Robótica. Robô Móvel. Processamento de Linguagem Natural. Acionamento Remoto.

ABSTRACT

Driving mobile robots could be via electrical or electronic control, manual or also by software programming using specific commands and language. However, a lay user likely would not get satisfactory results efficiently. An alternative, in this case, is driving the robot straight from Portuguese, in other words, using *natural language*.

With the proposal and evaluation of a system for supervision, control and operation of a robot by means of natural language in Portuguese, some recent technological advances in the fields of *robotics* and *artificial intelligence* are boarded with a practical view throughout a solution conceived focused in services.

Were embraced concepts such as the progression "data – information – knowledge", sufficient notions on knowledge representation, definition and structures of the natural language comparing to the artificial languages, the Natural Language Processing or Computational Linguistics, its implementation by means of deterministic and stochastic (statistical) methods, Maximum Entropy Models and how important is using ontologies for achieving expected results

In a practical implementation, it was proposed a solution which could serve as a functional proof of concept in using together these mentioned theories and technologies. No special resources were used, keeping in sight the objective of proposing an easily accessible solution, even based on new technologies. An integration software is presented for this proposed practical solution. Were evaluated: *functionalities* made available; *potentiality* of being created new ones; this architecture´s *capability of conforming* to the new features of a robot that come to be used; and diverse *human-machine interfaces* that could be used to drive services available.

Results were positive in more than three fourths of tested cases, including the variations allowed in natural language. Complex commands, with chaining of external referencing, straight instructions and automatic navigation had been evaluated with different writing styles and verbal times.

Keywords: Robotic Interface. Mobile Robot. Natural Language Processing. Remote drive.

LISTA DE ILUSTRAÇÕES

2.1.1 – Exemplo de Conceitos e Relacionamentos sobre animais. 11
2.1.2 – Exemplo de classificação: Árvore de Porfírio. 12
2.2.1 – Exemplo de relacionamento entre classes e subclasses do tipo "é_uma". 20
2.2.2 – Mapa Conceitual sobre Mapas Conceituais. 21
2.3.1 – Dado, Informação, Conhecimento e Inteligência. 24
2.8.1 – Distância "Euclidiana" entre dois pontos. 34
2.8.2 – Distância "*Manhattan*" entre dois pontos. 34
2.8.3 – Exemplo de mapa com pontos de origem e destino, obstáculo e uma rota. 36
2.8.4 – Arredores da origem avaliados, antes de fazer o primeiro passo. 37
2.8.5 – Arredores do ponto D3 avaliados, antes de fazer o próximo passo. 39
2.8.6 – Arredores do ponto C3 avaliados, antes de fazer o próximo passo. 40
2.8.7 – Ponto de destino foi atingido e incluído na "lista fechada". 41
2.8.8 – Caminho indicado pelos vetores de deslocamento, partindo do destino. 42
3.1.1 – Diagrama de um Processamento de Linguagem Natural genérico. 48
3.3.1 – Análise sintática de uma frase com estrutura em árvore. 56
3.3.2 - Análise *Top-Down* do exemplo e estrutura representada em árvore. 57
3.4.1 – Um exemplo de estrutura em árvore aplicada ao ATIS. 61
3.4.2 – Dependência entre Conceitualizações. 64
3.4.3 – Exemplo de representação usando Dependência Conceitual. 66
4.1.1 – Sistema Robótico esquematizado por blocos funcionais. 74
4.1.2 – Sistema Robótico esquematizado em detalhes. 75
4.1.3 - Sistema Robótico com esquematização generalizada. 77
4.2.1 – Visão geral da Interface em Linguagem Natural. 78
4.2.2 – Principais elementos da Interface em Linguagem Natural. 80
4.2.3 – Elementos de controle do sistema robótico. 84
4.2.4 – Elementos de controle do sistema robótico em "Modo de Teste". 86
4.6.1 – Visão geral do processamento das instruções em linguagem natural. 100
4.6.2 – Organização das funções de Processamento da Linguagem Natural. 102

4.6.3 – Visão geral da execução e do controle dos comandos. 103
4.6.4 – Organização das funções de Lógica de Controle e Comunicação. 103
6.1.1 – Questionários aplicados e questões numeradas de Q1 a Q25. 152
6.2.1 – Instrução imperativa e um caminho encontrado. 165
6.2.2 – Instrução afirmativa e um caminho encontrado. 167
6.5.1 – Teste de resposta para perguntas do tipo "Onde é/está/fica …". 177
6.5.2 – Teste de resposta para perguntas do tipo "Como chegar até…". 177
A.1.1 – Visão geral do robô Curumim. 205
A.1.2 – Roda omnidirecional e motorredutor do Curumim. 206
A.1.3 – Posicionamento do conjunto motorredutor-roda no Curumim. 206
A.1.4 – Vista superior do Curumim e indicação dos sensores. 207
A.1.5 – Vista frontal do Curumim e destaque para três sensores. 208
A.1.6 – Vista traseira do Curumim e destaque para dois sensores. 208
A.1.7 – Posição da câmera no Curumim e a câmera em detalhe. 209
A.1.8 – Receptor de vídeo para a câmera do Curumim. 210
A.1.9 – Detalhe da pinça incorporada no Curumim. 210
A.1.10 – Utilização da pinça. 211
A.1.11 – Placa de controle, ou placa mãe, do Curumim. 211
A.1.12 – Diagrama de blocos da Unidade Móvel robótica. 212
A.1.13 – Radiobase ou transceptor de comunicação sem fio. 213
A.1.14 – Baterias recarregáveis. 214
A.1.15 – Carregador das baterias e sua fonte de alimentação. 214
A.2.1 – Planta do espaço proposto para teste com usuários. 221
A.2.2 – Montagem do espaço para experiência com usuários – Vista 1. 222
A.2.3 – Montagem do espaço para experiência com usuários – Vista 2. 222
A.4.1 – Montagem do experimento para avaliar a comunicação sem fio. 227
A.4.2 – Outra vista do experimento para avaliar a comunicação sem fio. 227
A.5.1 – Tela inicial da miniaplicação de teste do *SharpNLP*. 231
A.5.2 – Resultado da função *SentenceDetect* pela miniaplicação do *SharpNLP*. 231
A.5.3 – Resultado da função *Tokenize* pela miniaplicação do *SharpNLP*. 232
A.5.4 – Resultado da função *PosTagger* pela miniaplicação do *SharpNLP*. 232
A.5.5 – Resultado da função *Chunker* pela miniaplicação do *SharpNLP*. 233
A.5.6 – Resultado da função *Parser* pela miniaplicação do *SharpNLP*. 233

A.5.7 – Resultado da função *NameFind* pela miniaplicação do *SharpNLP*. 234
A.5.8 – Tela inicial da aplicação de teste de comandos ao Curumim. 236
A.5.9 – Primeiro teste para o comando *Move*. 237
A.5.10 – Segundo teste para o comando *Move*. 237
A.5.11 – Terceiro teste para o comando *Move*. 238
A.5.12 – Quarto teste para o comando *Move*. 238
A.5.13 – Quinto teste para o comando *Move*. 239
A.5.14 – Sexto teste para o comando *Move*. 239
A.5.15 – Primeiro teste para o comando *Rotate*. 240
A.5.16 – Segundo teste para o comando *Rotate*. 240
A.5.17 – Terceiro teste para o comando *Rotate*. 240
A.5.18 – Quarto teste para o comando *Rotate*. 241
A.5.19 – Quinto teste para o comando *Rotate*. 241
A.5.20 – Sexto teste para o comando *Rotate*. 241
A.5.21 – Primeiro teste de comandos do robô. 244
A.5.22 – Segundo teste de comandos do robô. 245
A.5.23 – Terceiro teste de comandos do robô. 245
A.5.24 – Quarto teste de comandos do robô. 246

LISTA DE TABELAS

3.3.1 – Algumas categorias gramaticais constituintes de frases. 53
3.3.2 – Exemplo de gramática livre de contexto. 54
3.3.3 – Exemplo da aplicação de uma gramática livre de contexto. 55
3.3.4 – Exemplo de regras de gramática. 57
3.3.5 – Transcrição da gramática anterior. 57
3.3.6 – Passo a passo da análise *Top-Down* da frase "O gato caçou o rato". 58
3.3.7 – Análise *Bottom-Up* da frase "O gato caçou o rato". 58
3.4.1 – Ações Primitivas – ACT. 62
3.4.2 – Tempos Conceituais. 63
3.4.3 – Interpretação das Dependências entre Conceitualizações. 65
5.2.1 – Visão geral dos Esquemas de Casos e seus Papeis. 108
5.2.2 – Exemplo de verbos de ação úteis para o léxico. 110
5.2.3 – Exemplo de outras palavras úteis para o léxico. 111
5.3.1 – Blocos funcionais, Algoritmos e Procedimentos. 112
5.3.2 – Procedimento "Processa Linguagem Natural". 113
5.3.3 – Algoritmo "Separação de Orações pela Pontuação". 115
5.3.4 – Algoritmo "Identificação de Comandos e Parâmetros". 116
5.3.5 – Seção "Capturar Erros de Comando Ausente" do algoritmo. 116
5.3.6 – Seção "Avaliar Contexto da Oração" do algoritmo. 117
5.3.8 – Seção "Capturar Erros de Comando Ausente" do algoritmo. 119
5.3.9 – Algoritmo para o Caso do comando "Mover". 120
5.3.10 – Seção "Encontrar Sentido" do comando "Mover". 121
5.3.11 – Seção "Encontrar Distância" do comando "Mover". 122
5.3.12 – Seção "Encontrar Velocidade" do comando "Mover". 123
5.3.13 – Algoritmo para o Caso do comando "Rotacionar". 124
5.3.14 – Seção "Encontrar Sentido" do comando "Rotacionar". 124
5.3.15 – Algoritmo para o Caso do comando "Rotear". 125
5.3.16 – Seção "Encontrar Velocidade e Local" do comando "Rotear". 126

5.3.17 – Seção “Colocar Alvo na Planta” do comando “Rotear”. 127
5.3.18 – Algoritmo “Rumar Para”. 128
5.3.19 – Seção “Avaliar condições e preparar” do algoritmo “Rumar Para”. 129
5.3.20 – Seção “Transformar caminho em comando” do algoritmo “Rumar Para”. 130
5.3.21 – Algoritmo “Adição à Lista de Comandos”. 131
5.3.22 – Procedimento “Executar Comando”. 132
5.3.23 – Procedimento “Enviar Comando”. 134
5.3.24 – Procedimento “Executar Tudo”. 135
5.3.25 – Procedimento “Executar Parcial”. 136
5.3.26 – Procedimento “Executar Passo”. 136
6.1.1 – Tabulação das respostas às questões da figura 6.1.1. 152
6.1.1 (continuação) – Tabulação das respostas às questões da figura 6.1.1. 153
6.1.2 – Cálculos estatísticos sobre o Questionário Demográfico. 154
6.1.3 – Cálculos estatísticos para o Questionário sobre a experiência de uso. 154
6.1.4 – Extrato dos dados da Primeira Parte da experiência. 158
6.1.5 – Cálculos estatísticos dos resultados tabela 6.1.4. 158
6.1.6 – Extrato dos dados da Segunda Parte da experiência. 159
6.1.7 – Cálculos estatísticos dos resultados tabela 6.1.6. 160
6.2.1 – Instruções imperativas submetidas ao Sistema Robótico. 163
6.2.2 – Comandos e mensagens resultantes do texto da tabela 6.2.11. 163
6.2.3 – Outras instruções imperativas submetidas ao Sistema Robótico. 164
6.2.4 – Comandos e mensagens resultantes do texto da tabela 6.2.13. 164
6.2.5 – Instruções afirmativas submetidas ao Sistema Robótico. 165
6.2.6 – Comandos e mensagens resultantes do texto da tabela 6.2.15. 166
6.2.7 – Outra instrução afirmativa submetida ao Sistema Robótico. 166
6.2.8 – Comandos e mensagens resultantes do texto da tabela 6.2.17. 166
6.2.9 – Várias maneiras de escrever a mesma instrução. 167
6.2.10 – Comandos e mensagens resultantes do texto da tabela 6.2.19. 167
6.2.11 – Instrução na forma interrogativa submetida ao Sistema Robótico. 168
6.2.12 – Comandos e mensagens resultantes do texto da tabela 6.2.21. 169
6.2.13 – Outra instrução interrogativa submetidas ao Sistema Robótico. 169
6.2.14 – Comandos e mensagens resultantes do texto da tabela 6.2.23. 169
6.3.1 – Frases processadas diferentemente pelos modelos em Português. 170

6.5.1 – Perguntas submetidas ao Sistema Robótico em teste. 176
A.4.1 – Resultado da primeira avaliação da comunicação sem fio. 228
A.4.2 – Valores da tabela A.4.1 recalculados em porcentagem. 228
A.4.3 – Resultado da segunda avaliação da comunicação sem fio. 229
A.4.4 – Valores da tabela A.4.3 recalculados em porcentagem. 229

LISTA DE ABREVIATURAS E SIGLAS

ADJ	Adjetivo
ADV	Advérbio
ART	Artigo
ATIS	*Air Travel Information Service*
DET	Determinante
DIWK	*Data-Information-Knowledge-Wisdom*
	(Dado-Informação-Conhecimento-Sabedoria)
DRDC	*Defense Research and Development Canada*
ECD	*Explanatory Combinatorial Dictionary*
	(Dicionário Combinatório Explanatório)
F	Frase
GRACO	Grupo de Automação e Controle da Universidade de Brasília
HMM	*Hidden Markov Model* (Cadeia Oculta de Markov)
IFSP-SPO	Instituto Federal de São Paulo na capital
IFLN	Interface em Linguagem Natural
IFR	Interface com o Robô
IN	Interjeição
MaxEnt	*Maximum Entopy Model* (Modelo de Entropia Máxima)
MEMM	*Maximum Entropy Markov Model* (Modelo de Entropia Máxima de Markov)
MTM	*Meaning-Text Model* (Modelo Significado-Texto)
N	Substantivo
NLP	*Natural Languagem Processing*
NUM	Numeral
O	Oração
S	Sentença
SN	Sintagma Nominal
SV	Sintagma Verbal
PLN	Processamento da Linguagem Natural

PROP	Nome Próprio
PRP	Preposição
PUNC	Pontuação
V	Verbo
V-INF	Verbo no infinitivo
V-FIN	Verbo finito (tempo conjugado)
V-GER	Verbo no gerúndio
V-PCP	Verbo no particípio

SUMÁRIO

1. INTRODUÇÃO 1

1.1 Objetivos 4

1.2 Justificativas 4

1.3 Revisão Bibliográfica de Trabalhos Correlatos 6

1.4 Organização deste Trabalho 8

2. CIBERNÉTICA 10

2.1 Ontologia 10

2.1.1 Definição e origem 10

2.1.2 Aplicação e usos 13

2.1.3 Taxonomia, Vocabulário, *Thesaurus*, Glossário e Dicionário 14

2.2 Mapas Conceituais e Redes Semânticas 19

2.2.1 Representação do Conhecimento 21

2.2.2 Conhecimento e Ontologia 22

2.2.4 Tesauros ("*Thesaurus*") 22

2.2.5 Relação com a proposta deste trabalho 22

2.3 Inteligência Artificial e Aprendizado de Máquina 23

2.3.1 Dado, Informação, Conhecimento, Inteligência e Sabedoria 23

2.3.2 Relação com a Proposta deste Trabalho 24

2.4 Modelo de Entropia Máxima 25

2.4.1 Entropia Máxima 26

2.4.2 Construção de um Modelo de Entropia Máxima 27

2.5 Autômato de Estados Finitos 27

2.5.1 Conceituação 27

2.5.2 Relação com a proposta deste trabalho 28

2.6 Cadeia Oculta de Markov 28

2.7 Modelo de Markov de Entropia Máxima 29

2.8 Algoritmo de busca do melhor caminho 30

2.8.1 Iniciando a busca 36

2.8.2 Calculando os valores pela primeira vez 38
2.8.3 Prosseguindo com a busca 39
2.9 Teleoperação, Telerrobótica e Telepresença 42
2.10 Considerações Finais 43
3. LINGUÍSTICA COMPUTACIONAL 46
3.1 O Processamento da Linguagem Natural 46
3.1.1 Linguagem Natural 46
3.1.2 Processamento da Linguagem Natural 46
3.2 Processamento Léxico ou Morfológico 49
3.2.1 Ambiguidades Léxicas 51
3.3 Processamento Sintático 52
3.3.1 Gramática Livre de Contexto 53
3.3.2 Análise *Top-Down* e Análise *Bottom-Up* 56
3.4 Processamento Semântico 58
3.4.1 Gramática Semântica 60
3.4.2 Dependência Conceitual 62
3.4.3 Esquema de Casos 67
3.5 Processamento do Discurso e Processamento Pragmático 70
3.6 Considerações Finais 72
4. O SISTEMA ROBÓTICO PROPOSTO 73
4.1 Visão Geral e Arquitetura 73
4.1.1 Visão detalhada do sistema proposto 75
4.1.2 Esquematização generalizada do sistema 76
4.2 A Interface em Linguagem Natural 77
4.2.1 Elementos que constituem a interface 79
4.2.2 Visão geral da forma de utilização da IFLN 86
4.2.3 Detalhamento da forma de utilização 87
4.2.4 Visão geral do funcionamento 89
4.3 O *Software* de Inteligência e Controle 90
4.3.1 Funções de Processamento da Linguagem Natural 90
4.3.2 Funções da Lógica de Controle do Robô 95
4.3.3 Funções de Controle da Comunicação entre o Sistema e o Robô 98
4.4 A Interface com o Robô 98

4.5 O Robô 99
4.6 Funcionamento do Sistema Robótico Proposto 99
4.7 Considerações Finais 104
5. O DESENVOLVIMENTO DO SISTEMA ROBÓTICO 106
5.1 Planejamento e Restrição de Contexto 106
5.2 Planejamento e Esquema de Casos 108
5.2.1 Abstrações agregam utilidade 108
5.2.2 Planejamento do Léxico 110
5.3 Algoritmos e Procedimentos Implementados 111
5.3.1 Procedimento "Processa Linguagem Natural" 112
5.3.2 Algoritmo "Separação de Sentenças" 113
5.3.3 Algoritmo "*Tokenização* de Sentença" 113
5.3.4 Algoritmo "Marcação da Função Sintática" 114
5.3.5 Algoritmo "Separação de Orações pela Pontuação" 114
5.3.6 Algoritmo "Identificação de Comandos e Parâmetros" 115
5.3.7 Algoritmo "Rumar Para" 127
5.3.8 Algoritmo "Adição à Lista de Comandos" 131
5.3.9 Procedimento "Executar Comando" 131
5.3.10 Procedimento "Enviar Comando" 133
5.3.11 Procedimento "Executar Tudo" 135
5.3.12 Procedimento "Executar Parcial" 135
5.3.13 Procedimento "Executar Passo" 136
5.4 A elaboração da interface 137
5.4.1 Observações sobre o desenvolvimento 139
5.5 A elaboração do *software* 139
5.5.1 O processamento sintático 140
5.5.2 O processamento semântico 142
5.5.3 Planta do local e algoritmo de busca de rota 144
6. TESTES E RESULTADOS 149
6.1 Testes do Sistema Robótico Proposto com Usuários Leigos 149
6.1.1 O ambiente da experiência 150
6.1.2 Padronização da experiência 150
6.1.3 Apresentação e avaliação dos resultados 151

6.1.4 Observações do pesquisador 161
6.2 Testes das Funcionalidades do Sistema Robótico Proposto 162
6.2.1 Teste 1 – Requisições na forma afirmativa 162
6.2.2 Teste 2 – Requisições na forma interrogativa 168
6.3 Dificuldades encontradas 169
6.4 Limitações 171
6.4.1 Verbos como base da teoria dos Esquemas de Casos 171
6.4.2 Mudança na ordem dos comandos 171
6.4.3 Sobre as quantidades numéricas 172
6.4.4 Novos locais de referências e memória de contexto 172
6.4.5 Uso de coordenadas 172
6.4.6 Uso de correferências 173
6.4.7 Busca de caminho à partir de ponto futuro 173
6.4.8 Uso de instruções negativas 173
6.4.9 Uso combinado de comandos 174
6.4.10 Nome das referências 176
6.5 Observações Finais 176
7. DISCUSSÃO, CONCLUSÃO E TRABALHOS FUTUROS 180
7.1 Discussão 180
7.1.1 Contextualização 180
7.1.2 Realização do Sistema Robótico Proposto 180
7.1.3 Interoperabilidade 181
7.1.4 Foco em serviços 182
7.1.5 Possibilidades de aplicação das soluções encontradas 183
7.2 Conclusão 183
7.3 Contribuições 184
7.4 Trabalhos Futuros 185
7.4.1 Esclarecimento 185
7.4.2 Multiusuário e multirrobô 185
7.4.3 Utilização de números na forma literal 186
7.4.4 Instruções de mudança na ordem ou sequência dos comandos 186
7.4.5 Deslocamento por meio de coordenadas 186
7.4.6 Memória de contexto e resolução de correferências 186

7.4.7 Aprendizado pela Interface em Linguagem Natural 187
7.4.8 Algoritmos de Busca Heurística de Solução 187
7.4.9 Busca de Rota ou Caminho a partir de um ponto futuro 187
7.4.10 Orações com sujeito oculto 188
7.4.11 O dicionário de palavras raras do *SharpNLP* 188
7.4.12 Capacidade de responder perguntas 188
7.4.13 Novo *Corpus* de treinamento 189
7.4.14 Ferramenta para edição do léxico 189
7.4.15 Módulo de ajuda ao usuário 190
8. REFERÊNCIAS BIBLIOGRÁFICAS 191
A. APÊNDICE 204
A.1 O Robô Curumim 204
A.1.1 Sistema de Locomoção 205
A.1.2 Sensores infravermelho 207
A.1.3 Câmera Colorida 208
A.1.4 Pinça acionável 210
A.1.5 Eletrônica embarcada 211
A.1.6 Radiobase e comunicação sem fio 213
A.1.7 Baterias recarregáveis e carregador 213
A.1.8 Possibilidades de controle do Curumim 215
A.2 Metodologia de Teste 218
A.2.1 Recursos necessários 218
A.2.2 Medições e registros 218
A.2.3 Instruções ao pesquisador 218
A.2.4 Tarefa da segunda parte da experiência 220
A.3 Modelos de Documentação e Questionário de Teste 223
A.3.1 Descritivo da experiência e Termo de esclarecimento e livre consentimento 223
A.3.2 Questionário demográfico 224
A.3.3 Questionário sobre a experiência de uso 225
A.4 Teste da comunicação com o robô 226
A.4.1 Objetivo do experimento 226
A.4.2 Descrição do experimento 226
A.4.2 Resultado do experimento 228

A.5 Testes e resultados durante o desenvolvimento 230
A.5.1 Teste 1 – Conhecendo as funções do *SharpNLP* 230
A.5.2 Teste 2 – Primeira tentativa de controle do robô 242
A.5.3 Teste 3 – Os modelos em Português para o *SharpNLP* 247

1. INTRODUÇÃO

Quando as transformações pelas quais o planeta e o mundo passaram são observadas com cuidado, desde a Revolução Industrial, é possível perceber que não foi somente o Ser Humano, com sua capacidade criativa, o único ator dessas transformações. Bem antes disso, dentre os eventos que a história registrou e que temos conhecimento hoje, um dos mais antigos e dignos de menção é a criação de *Ctesíbio de Alexandria*, em cerca de 250 a.C.: um relógio d'água que dispunha de recurso de autocontrole para manter o fluxo constante e previsível. Essa invenção revolucionou o conceito do que um artefato seria capaz de realizar (RUSSEL; RUSSEL, 2004). Então, principalmente com a já mencionada Revolução Industrial, surge a importante figura das máquinas. Mas elas já não eram nenhuma novidade nesta época, pois foram concebidas e construídas, várias delas, muito tempo antes. Um importante avanço do período, responsável por tornar as máquinas tão importantes hoje, foi o ressurgimento do controle. Com o *Fly-Ball* de James Watt (OGATA, 2003), era possível controlar a força mecânica das máquinas a vapor com pouca ou nenhuma intervenção humana, ou seja, não era necessário designar alguém para ajustar uma válvula toda vez que a rotação mudasse: o ajuste era realizado automaticamente por um dispositivo apropriado da própria máquina. E os trabalhadores poderiam dedicar-se mais aos trabalhos de maior elaboração.

Com o passar dos anos, a tecnologia se desenvolveu e o conhecimento sobre o assunto se aprofundou. As ideias e os conceitos se solidificaram, novas tecnologias surgiram e tornou-se possível a abstração das aplicações e das funções até o nível do *software*, num estado de plena abstração ou ausência de presença física.

O *software* abriu as portas para a representação das ideias de forma mais pura e imediata. O mundo virtual, criado pelo *software*, está mais próximo do mundo das ideias. Ele pode ser moldado por elas com pouco esforço e mais facilidade que o mundo físico.

Ainda hoje é possível ver máquinas operando sem um *software* puro e explícito, ainda que ele esteja implícito na funcionalidade dos mecanismos de controle. Contudo, os recursos do *software* digital, aliados aos microprocessadores e

microcontroladores, se desenvolveram ao ponto de ser possível dotar as máquinas de recursos inteligentes sem precedentes. De fato, a inteligência artificial, um dos grandes avanços do século vinte, hoje pode ser encontrada em brinquedos, sendo também que peças basilares de *software* podem ser encontradas praticamente sem custo nas comunidades livres ou de código aberto. Foi o *software* que permitiu o desenvolvimento da inteligência artificial e também maximizou a potencialidade de recursos e benefícios do controle digital das máquinas.

Por outro lado, o *software* é abstrato e não possui existência real no mundo físico. As máquinas, dotadas de seus múltiplos transdutores, é que são capazes de interagir com as forças da Natureza e interferir no mundo físico. Elas é que são colocadas para fazer o que nós, Seres Humanos, não queremos ou não conseguimos fazer. Nos ambientes extremos ou nas condições de trabalho onde nossa própria máquina orgânica não pode existir sem danos severos, são colocadas as máquinas no lugar.

E assim, o planeta e o mundo se transformaram numa riqueza de possibilidades sem precedentes. O Ser Humano, agindo em equipe, foi capaz de assumir sua imagem e semelhança. A Natureza, que também criou o Ser Humano, agora retorna pelas forças das máquinas, aparentemente como criatura dos Humanos, e lado a lado se tornam mutuamente responsáveis por suas próprias transformações.

Dos avanços surgidos após a Revolução Industrial, alguns deles mais recentes e tecnológicos, nas áreas da *robótica*, da *inteligência artificial* e das *redes de comunicação* serão abordados de forma prática durante esta dissertação. O potencial gerador de aplicações dessas três tecnologias, ao serem exploradas juntas, permite conceber o surgimento de máquinas dotadas de capacidades verdadeiramente surpreendentes para a maioria dos Seres Humanos leigos. Somente com duas delas – a *robótica* e a *inteligência artificial* – é possível conceber máquinas mais próximas do que da definição de um Ser Humano. A interação entre Humanos e Máquinas Inteligentes tende a se desenvolver mais para a forma Humana de se relacionar. São as máquinas que se adaptarão a nós, uma vez que nós não nos adaptamos muito bem a elas.

Porém, a terceira dessas três tecnologias abre campo para uma vasta gama de possibilidades impossíveis ao Ser Humano. As redes de comunicação permitem às máquinas interagir fisicamente em locais distantes, ou mesmo em múltiplos locais. Se

hoje é possível falar em computação ubíqua[1] (ARAÚJO, 2003), é possível falar também em robótica ubíqua (JONG; YONG; KANG, 2004) (KIM; LEE; KIM, 2007), teleoperação e telerrobótica (Álvares; Romariz, 2002). Ou seja, é possível conceber uma inteligência artificial ubíqua, pervasiva e fisicamente ativa nos ambientes, quer sejam eles habitados por Seres Humanos ou não.

Como forma de iniciar os primeiros passos em um horizonte potencial como esse, a proposta deste trabalho é trazer e avaliar uma solução para acionar um robô com linguagem natural: a forma humana de comunicação.

Neste contexto, uma possível alternativa para interface entre os seres humanos e as máquinas é dotá-las com capacidade de compreender a linguagem utilizada pelos seres humanos para comunicação e descrição de tarefas cotidianas. Como observam Levine, Drang e Edelson (1998), a criação de um sistema de comunicação entre pessoas e computadores é chamada de Processamento de Linguagem Natural - PLN, onde seu objetivo final é permitir que o usuário interaja com o computador usando a linguagem humana. A máquina é dotada de inteligência e controle suficientes para interpretar o que foi dito, executar o trabalho conforme instruído e reportar os resultados alcançados.

A Cibernética é uma ciência interdisciplinar que aborda sistemas regulatórios, estudo dos sistemas de controle e se aplica tanto aos sistemas físicos quando aos sociais. A palavra Cibernética foi registrada pela primeira vez por Platão, na sua obra As Leis, para designar a governança das pessoas. Em 1834, *André Marie Ampère* usou a palavra *Cybernétique* referindo-se às ciências governamentais no seu sistema de classificação do conhecimento humano (CIBERNETICS, WIKIPEDIA). E no final dos anos de 1940, *Norbert Wierner* organizou conferências com *Warren McCulloch*, *Walter Pitts* e *John Von Neumann* para explorar modelos matemáticos de cognição. Seu livro "*Cybernetics, or the study of control and communication in the animal and the machine"* (Cibernética, ou o estudo do controle e da comunicação no animal e na máquina - 1948) influenciou pesquisadores e abriu portas para as máquinas dotadas de inteligência artificial (HEYLIGHEN; JOSLYN, MEYERS, 2001).

[1] Computação Ubíqua: conceito onde o processamento acontece de forma presente no ambiente comum humano, em qualquer objeto, de forma a estar sempre presente e disponível, convenientemente, não sendo possível precisar um único local aonde ela acontece ou se manifeste.

1.1 OBJETIVOS

O principal objetivo deste trabalho é o de idealizar e construir um sistema de supervisão, operação e controle de robôs através da interação entre humanos e máquina via linguagem natural em aplicações direcionadas ao uso da tecnologia com foco em serviços. E envolve principalmente *software*, dadas as suas vantagens em relação aos sistemas físicos.

Um dos pontos focais que orienta o desenvolvimento é ter como resultado serviços úteis. Isso quer dizer que o sistema (*software* + *hardware*) deverá dispor de alguma utilidade prática para realização de alguma tarefa específica por um robô através de linguagem natural.

Outro ponto focal da implementação, o da facilidade de comunicação entre Humanos e Máquinas via linguagem natural, poderá permitir que as metodologias e tecnologias que forem estudadas sirvam para um enriquecimento tecnológico e bibliográfico, possibilitando o uso da sistemática desenvolvida em outras aplicações.

Ao lado disso, a integração futura com diferentes ou novos dispositivos automatizados, junto ao sistema, poderá aumentar a abrangência de atividades e funções que podem ser desenvolvidas pelo(s) robô(s). Assim, o sistema que for idealizado deverá permitir a interoperabilidade com outros mecanismos.

1.2 JUSTIFICATIVAS

As mudanças que podem ser observadas no mundo são decorrentes de dois principais atores: Seres Humanos e Máquinas. A velocidade em que as mudanças têm ocorrido é aumentada com o próprio desenvolvimento e à medida que as melhorias são realizadas. Ou seja, as mudanças provocam melhorias e as melhorias ocasionam mudanças com mais agilidade e facilidade.

As máquinas, inicialmente idealizadas por Seres Humanos para auxiliarem no trabalho, hoje são usadas para produzir outras máquinas. Como no caso da indústria automobilística, robôs são usados para montar carros e no caso da microeletrônica, computadores feitos de *microchips* são usados para projetar e aperfeiçoar novas versões de *microchips*.

Esse fenômeno de metadesenvolvimento ou metaevolução das máquinas não ocorre por si só, mas ocorre graças à intervenção criativa e organizadora humana. Essa intervenção somente é possível mediante uma interação entre ambos os atores.

Organização pressupõe controle e o fenômeno do controle, direção, gestão ou governança envolvendo Humanos e Máquinas é uma questão tradicional abordada pela cibernética.

Assim, nesse constante acelerar das mudanças do mundo, é presumível que a interação entre Homens e Máquinas seja observada cada vez mais de forma frequente e aprofundada. E como forma de isso ser também desenvolvido, na parte em que lhe compete, o contato precisa se tornar mais simples, amigável, intuitivo e conveniente. Ou seja, cada vez mais natural para os Seres Humanos. Pelo menos poderá ser observado assim enquanto estes últimos forem os mais inteligentes e criativos.

Este trabalho contribui ao reunir algumas referências significativas, avaliações sobre suas utilidades e alguns significados práticos durante uma aplicação real. Ao lado disso, é feita uma abordagem utilizando o Português como língua natural, uma vez que é a sexta mais utilizada no mundo (LEWIS, 2009). Pesquisar, estudar e avaliar algumas teorias e tecnologias que podem ser utilizadas para desenvolver a interação entre Humanos e Máquinas, através da construção de um sistema prático, trouxe à luz dos fatos algumas possibilidades, alguns problemas de ordem prática e alguns possíveis desenvolvimentos e estudos futuros.

A importância do *software* para concepção e desenvolvimento do sistema proposto pode ser compreendida quando se observa sua conveniência e flexibilidade em comparação com os dispositivos físicos. Ele é imaterial, abstrato e mais próximo da natureza das ideias em si, sendo maleável, flexível e dinâmico o suficiente para que seja moldado facilmente ao que se pretende, com bem menos esforço e tempo do que um dispositivo puramente físico. Isso significa menores custos. Também é poderoso em termos dos recursos oferecidos, como a possibilidade de realizar comparações, de armazenar parâmetros e informações, de executar cálculos e de elaborar resultados a partir das entradas alimentadas.

Assim, o *software* é a forma que permite o maior grau de inteligência e flexibilidade de recursos necessários para o processamento da linguagem humana.

O desenvolvimento do sistema envolve também equipamentos de *hardware*. O conhecimento dos recursos oferecidos por estes dispositivos é necessário para

determinar os limites alcançáveis do que o sistema é capaz executar em termos físicos, como a comunicação entre Humanos e Máquinas ou o alcance da atuação no ambiente: computadores, interfaces, dispositivos de comunicação e o robô.

1.3 REVISÃO BIBLIOGRÁFICA DE TRABALHOS CORRELATOS

A ideia de permitir que máquinas compreendam a linguagem natural humana, especialmente os robôs, não é nova e não é difícil encontrar histórias de ficção que abordam o tema, como também é assunto de estudo científico há bastante tempo.

A seguir serão comentados alguns trabalhos científicos pesquisados e encontrados que parecem se aproximar mais dos assuntos abordados por este trabalho, ou seja, comando e controle de robôs móveis por linguagem natural orientado a serviço.

Selfridge e Vannoy (1986) apresentam uma interface em linguagem natural capaz de compreender, aprender e executar comandos em um sistema de robô fixo, usando um computador VAX 11/780 para interpretação da linguagem natural, um computador Apple II+ para controlar manipuladores Microbot Minimover-5 de cinco graus de liberdade, um dicionário de cerca de 50 palavras e duas câmeras (uma de alta e outra de baixa resolução). Como o processamento da linguagem natural adotado é de grande complexidade, a capacidade de interpretação foi de alto nível, mesmo que a velocidade de desempenho geral da composição seja modesta, segundo os autores.

Spiliotopoulos *et al* (2001) reportam resultados obtidos com experimentos a cerca da interação entre robôs móveis e humanos por meio de diálogos falados. Trata-se de um robô móvel destinado a oferecer assistência em hospitais. Focam no sistema de gerenciamento de diálogo e abordam brevemente as técnicas utilizadas.

Nair e Prasad (1997) apresentam uma interface capaz de extrair conteúdo semântico sobre comandos executáveis por um robô. Por meio de um sistema de monitoração de sensores e um resolvedor de lógica *fuzzy*, é capaz de executar os comandos em tempo real, permitindo aos usuários interagirem com um robô em tempo real e também reconfigurar a interface para novos robôs.

No artigo, Lopes *et al* (2003) descrevem a arquitetura e habilidades de um protótipo de robô inteligente projetado para servir refeições em recepções ou agir como recepcionista. Segundo os autores, o protótipo é resultado de esforço de integração

entre as principais dimensões da inteligência: Comunicação, Ação, Decisão e Aprendizado. O artigo focaliza a capacidade de comunicação multimodal do protótipo.

Para investigar uma arquitetura usada em sistemas de comunicação por diálogo falado capaz de extrair procedimentos executáveis por robôs á partir de instruções de usuários, Lauria *et al* (2002) aborda a questão da característica da comunicação humana de expressar as tarefas complexas em pedaços. Isto requer que o sistema seja capaz de mapear os pedaços de instruções entre primitivas de procedimentos pré-programadas. Propõem uma PSL (*Procedure Specification Language* ou Linguagem de Especificação de Procedimento) e a demonstram no caso de uma instrução de rota.

Buscando simplificar a comunicação humana com robôs, e assim diminuir o tempo necessário para de interação para que resultados efetivos possam ser obtidos, Dahl, Fall e Thomas (1995) demonstram uma solução que combina duas visões. Ou seja, utilizam um mecanismo de dedução de alto nível para produzir um plano geral ao mesmo tempo em que se executa uma verificação dinâmica de baixo nível, distribuída com o programa robótico, para fazer a interface dinâmica com as ações do robô e as informações relativas ao seu desempenho. O artigo examina o problema particular da representação da linguagem natural no contexto do robô. Apresenta um sistema lógico conciso especialmente concebido para acomodar sentenças imperativas enviadas ao robô. Denominado de CL (*Command Language* ou Linguagem de Comando), também é apresentado um analisador que traduz comandos em Inglês para CL. Os autores esclarecem que a CL serve não só para a representação interna da linguagem, mas também como o objetivo é expresso mo planejador de alto nível do robô.

De forma a tornar a interação entre robôs e humanos acessível e natural para ambos e visando minimizar erros e desvios de interpretação, um modelo de léxico é proposto por Zelek (1997) para especificar comandos a um robô móvel executar tarefas de navegação. Deriva do conjunto semântico humano e o léxico de comando de tarefa consiste de palavras para verbo, destino, direção e velocidade. Não é intenção do autor que o modelo constitua-se nos únicos comandos capazes de reconhecimento pelo robô, mas que seja uma linguagem mapeável internamente nos construtores de planejamento e controle.

Com foco no que seria mais natural para usuários humanos, considerando que a maioria dos sistemas robóticos experimentais envolvendo linguagem natural não

são sistematicamente testados com usuários humanos não familiarizados com o sistema, Moratz e Tenbrik (2002) apresentam um cenário simples onde humanos se referem a objetos por meio de sua localização ao invés da descrição de suas características. O robô então utiliza um modelo de referências espaciais para interpretar as instruções linguísticas. Os relatos dos experimentos com usuários leigos mostram a adequação do modelo para alcançar a referência de localização. Também mostra como a abordagem pode ser expandida para tarefas espacialmente mais complexas do ponto de vista da interação entre humanos e robôs.

Song *et al* (2004) abordam o contraste existente entre a linguagem natural muito abstrata e a necessidade dos comandos para robôs móveis serem muito específicos e precisos. Ou seja, o processamento da linguagem natural e os sistemas de navegação apresentam um vazio a ser preenchido, qual é a proposta da pesquisa dos autores. O artigo mostra a inserção de um *Middleware* (*software* intermediário) padrão para melhorar a portabilidade e a eficiência do sistema. Nos experimentos, utilizam comandos como "Por favor, poderia ir até a porta." ao passo que o robô responde com "Estou indo para a porta... Eu estou na porta.". Demonstram que suas pesquisas apresentam resultados portáveis e reutilizáveis com a expansão do experimento para controle de um robô manipulador.

1.4 ORGANIZAÇÃO DESTE TRABALHO

Uma vez que os assuntos abordados neste trabalho são vastos e inexistem obras de referência que abordem a totalidade ou a maior parte do que aqui se procura apresentar, constituiu-se desafio adicional reunir e resumir as teorias, conceitos e tecnologias para permitir o entendimento necessário do que é feito ou proposto.

Cada capítulo se inicia com um preâmbulo que contextualiza o leitor sobre os assuntos que serão abordados e, ao final, foram incluídas seções com considerações finais para resumir o conteúdo abordado, considerando sua utilidade ou aplicação.

O capítulo 2 (Cibernética) inicia o conteúdo teórico com subsídios necessários para fomentar elementos do raciocínio cibernético no qual se baseia este trabalho.

O capítulo 3 (Linguística Computacional) aborda a complexidade, estrutura e teorias sobre a Linguagem Natural e seu processamento do ponto de vista computacional, tendo-se em vista os resultados que se deseja.

O capítulo 4 (Sistema Robótico Proposto) esclarece sobre a composição, funcionamento e recursos daquilo que se constitui em prova de conceito funcional e ponto focal da maioria dos esforços deste trabalho.

O capítulo 5 (Desenvolvimento do Sistema Robótico) lança comentários e esclarecimentos sobre os problemas encontrados, as alternativas encontradas, as soluções propostas, as decisões e compromissos assumidos durante a elaboração do Sistema Robótico Proposto. Também, nesta seção faz-se um detalhamento dos procedimentos e algoritmos implementados para permitir seu funcionamento.

O capítulo 6 (Testes e Resultados) documenta os testes, os resultados, as análises e as considerações mais significativas que foram obtidos e formalizados através da experimentação com o Sistema Robótico proposto.

O capítulo 7 (Discussão, Conclusão e Trabalhos Futuros) encerra o presente trabalho lançando vistas, críticas e opiniões sobre os resultados obtidos em relação aos objetivos propostos.

2. CIBERNÉTICA

A ciência que estuda os princípios abstratos de organização em sistemas complexos denomina-se Cibernética. Ela não se preocupa com a composição dos sistemas (do que são feitos), mas como funcionam, como utilizam as informações, os modelos e controlam as ações para se direcionarem e manterem seus objetivos à medida que experimentam várias perturbações. Sendo inerentemente transdisciplinar, o raciocínio cibernético pode ser aplicado para compreender, modelar e projetar sistemas de qualquer tipo: físico, tecnológico, biológico, social ou uma combinação deles (HEYLIGHEN; JOSLYN; MEYERS, 2001).

Neste capítulo serão apresentados alguns conceitos, teorias e tecnologias úteis para elaboração do sistema robótico objetivado por este trabalho. A organização das partes que compõem o sistema robótico pode tanto ser decorrente de uma utilização direta dos recursos ou tecnologias como também utilizá-las como embasamento para algum tipo de desenvolvimento ou então utilizá-las por meio de abstração de suas possibilidades, num enfoque de utilização procurando por recursos mais convenientes.

Posteriormente, nos capítulos 4 e 5, com a apresentação do sistema robótico serão tecidos comentários esclarecendo como foram utilizados esses conceitos, teorias e tecnologias visando obter os resultados almejados.

2.1 ONTOLOGIA

2.1.1 DEFINIÇÃO E ORIGEM

A palavra Ontologia origina-se de dois termos gregos: "*onta"*, que significa entes e "*logos"*, que pode significar teoria, discurso ou palavra. Assim, Ontologia pode ser entendida como estudo, teoria ou conhecimento dos entes ou das coisas que existem. Porém, o termo é utilizado de forma um pouco diferente conforme a área de conhecimento que o utiliza (CASTRO, 2008).

Segundo Chen (2009), para a Ciência da Informação, uma Ontologia é uma forma de representar o conhecimento. Ontologia é uma especificação da conceitualização (GRUBER; 1993). Seus principais componentes são:

- Conceitos (por exemplo: humano, animal, comida, mesa, filme etc.);
- Instâncias (por exemplo: Angelina Jolie é uma instância do conceito "pessoa");
- Propriedades (por exemplo: um humano possui as propriedades gênero, altura, peso, pai, mãe etc.);
- Relações (por exemplo: o IFSP-SPO está localizado na cidade de São Paulo);
- Papéis (por exemplo: se alguém é casado então se presume que possua um esposo ou uma esposa).

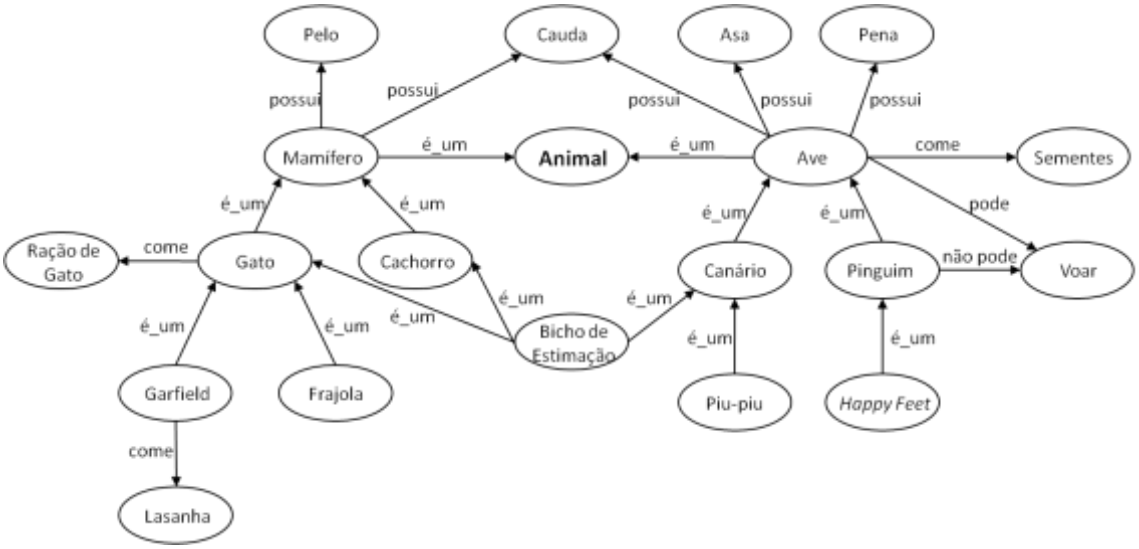


Figura 2.1.1 – Exemplo de Conceitos e Relacionamentos sobre animais.
Fonte: (SWIGGER, 2010)

Na filosofia, Aristóteles procurou estudar a maneira como entendemos e diferenciamos ou não a existência de algo. Ele utilizava a expressão “categoria” para classificar as coisas e adotou também a palavra “*differentia*” (diferença) para se referir as variações peculiares observadas em uma mesma categoria (ALMEIDA, *et al*., 2003). A expressão Ontologia foi cunhada por R. Goclenius para sua obra intitulada *Lexicon Philisophicum* em 1613 (CASTRO, 2008), porém numa abordagem diferente.

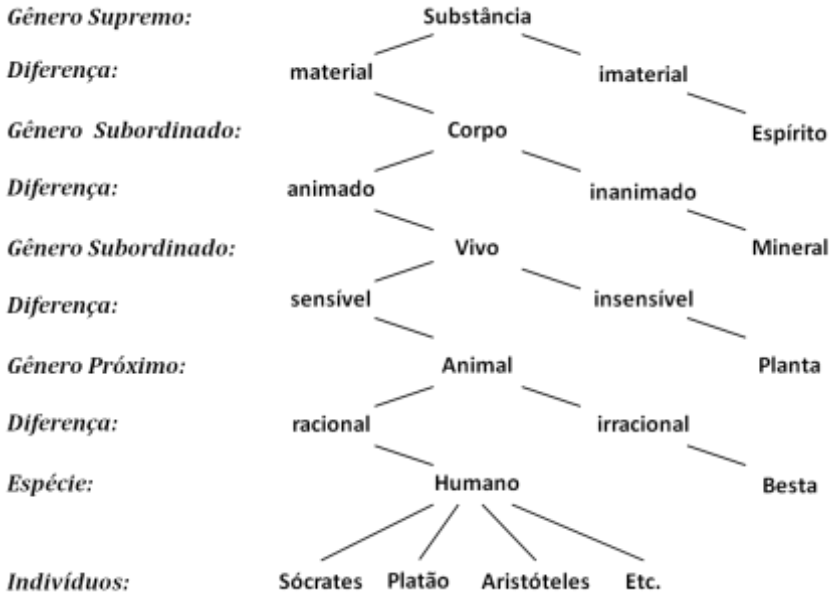


Figura 2.1.2 – Exemplo de classificação: Árvore de Porfírio.
Fonte: http://www.jfsowa.com/talks/ontology.html

Borst (1997) a definiu: "Uma ontologia é uma especificação formal e explícita de uma conceitualização compartilhada". Neste caso ele entende "formal" como sendo inteligível por computadores; a "especificação explícita", como sendo conceitos, relações, propriedades, funções, delimitações e regras; a ideia de "compartilhado", como sendo oriundo de consenso; e o termo "conceitualização", como sendo a representação do que acontece, ou seja, modelos.

O modelo de organização dos conceitos em categorias proposto por Aristóteles usufrui de certa propriedade natural de associação que se origina da memória semântica na qual uma subcategoria é considerada como parte de uma categoria quando possui propriedades (ou predicados, atributos) que são a essência do conceito (AMORETTI; TAROUCO, 2000).

Recentemente, a palavra Ontologia tomou maior vulto em estudos nas áreas de Inteligência Artificial, Linguística Computacional e Teoria de Dados sendo objeto de pesquisa também para Representação do Conhecimento, Engenharia do Conhecimento, e Arquitetura de Sistemas Baseados em Agentes e Análise Orientada a Objetos (CASARE, 2005).

A definição de Ontologia é diversificada na literatura de inteligência artificial e muitas se contradizem. Porém, para um entendimento inicial para propósitos didáticos

pode ser o proposto por Noy e McGuinness (2001): uma ontologia é uma descrição formal de conceitos no domínio do discurso.

Ao se estabelecer categorias para as coisas e propor conceitos para o que existe em um dado domínio é o processo de elaborar ontologias. Uma ontologia é um "catálogo de tipos de coisas" que supostamente existem no dado domínio, sob a óptica de observação de uma pessoa que faz uso da linguagem (ALMEIDA, et al., 2003).

Quando associada a um conjunto de instâncias de classes constituirá uma base de conhecimento. Segundo esses autores, há linha tênue onde a ontologia termina e a base de conhecimento começa. As classes são o foco da maioria das ontologias e elas, as classes, descrevem conceitos em um domínio (de conhecimento) (NOY; MCGUINESS, 2001).

Estabelecer um conceito é o mesmo que definir regras que delimitam como entender um ente, bem como descrever este ente em linguagem natural. Portanto, uma ontologia é uma associação entre o significado da descrição de uma coisa com seu nome, através das regras que foram formalizadas. Isso é o mesmo que as afirmativas observadas nas teorias lógicas (CASARE, 2005).

2.1.2 APLICAÇÃO E USOS

Segundo Noy e McGuinness (2001), recentemente, o desenvolvimento de ontologias – especificação formal e explícita de termos num domínio e as relações entre eles – tem saído do ambiente dos laboratórios de inteligência artificial para os escritórios dos especialistas nesses domínios. Uma ontologia define vocabulários comuns para pesquisadores que precisam compartilhar informação. Porém, isso inclui definições do conceito básico e os relacionamentos entre elas que sejam interpretáveis por máquinas. E algumas das razões que levam ao desenvolvimento de uma ontologia são:

- Compartilhar entendimento comum da estrutura da informação entre pessoas e agentes de *software*;
- Permitir reutilização do domínio de conhecimento;
- Explicitar as hipóteses ou suposições;
- Separa o domínio de conhecimento do conhecimento operacional;
- Analisar o domínio de conhecimento.

Importantes filósofos, como Aristóteles e Wittgenstein, abordaram as questões da existência das coisas e uma forma lógica de categorizá-las para organizar o pensamento sobre o mundo. Nas diversas correntes e escolas filosóficas, as visões sobre o assunto são diferentes. O termo "ontologia" nasceu dentro da filosofia, da metafísica, como o estudo da natureza do ser e da sua existência, se expandiu e hoje permeia outras áreas do conhecimento, como a Linguística e as ciências da Computação e da Informação (KASAMA; ZAVAGLIA; ALMEIDA, 2010).

O uso de ontologias se mostrou uma forma eficiente de representar conceitos de forma semanticamente relacionada. As ontologias são capazes de representar e descrever um conhecimento de maneira formal o suficiente para permitir a elaboração de inferências sobre a informação estruturada através da aplicação de lógicas. Sua utilidade se mostrou tanto nos sistemas de bancos de dados como nos de Processamento da Linguagem Natural (KASAMA; ZAVAGLIA; ALMEIDA, 2010).

Segundo os mesmos autores, o uso do conceito de ontologias representa uma forma de estruturar o conhecimento e que possibilita:

a) conceitos estruturados com base em classes, subclasses e herança de conceitos;
b) concordância com um conceito cada vez mais utilizado para descrição de domínios de conhecimento;
c) estruturação formal o suficiente para permitir que recursos disponíveis na forma linguística possam ser tratados computacionalmente, inclusive com possibilidade de realização de inferências automáticas a partir de regras lógicas pré-determinadas.

2.1.3 TAXONOMIA, VOCABULÁRIO, THESAURUS, GLOSSÁRIO E DICIONÁRIO

Algumas palavras apresentam definições semelhantes, porém ligeiramente diferente do que foi visto sobre ontologia pouco antes. Portanto, parece ser prudente passar a avaliar um pouco melhor o significado de cada uma delas.

Iniciando por procurar o significado no dicionário, será possível encontrar (AULETE DIGITAL, 2011 – exceto quando citado diferentemente):

Léxico: repertório de palavras de uma língua, de um texto, usadas por um autor ou por uma escola ou movimento literário, sendo, portanto, um sinônimo de “vocabulário” ou mesmo de “dicionário”.

Vocabulário: conjunto dos termos característicos de uma atividade ou campo do conhecimento (por exemplo, vocabulário do futebol, vocabulário psicanalítico); conjunto das palavras usadas por um autor em sua obra, por uma faixa etária, um grupo social ou de interesses etc. (por exemplo, vocabulário camoniano, vocabulário jovem, vocabulário dos sambistas); conjunto das palavras que uma pessoa conhece; dicionário de particularidades das palavras, que não inclui necessariamente definição de acepções (por exemplo, vocabulário ortográfico, vocabulário etimológico); conjunto dos símbolos usados para codificação de operações e instruções numa linguagem de programação; genericamente é o conjunto dos vocábulos de uma língua e um sinônimo de léxico.

Glossário: vocabulário que vem anexo a uma obra para explicar palavras e expressões técnicas, regionais ou pouco usadas contidas no texto; elucidário de termos técnicos (por exemplo, glossário de termos médicos); pequeno léxico de termos obscuros ou pouco conhecidos posto no final de uma obra para elucidar o significado dessas palavras (por exemplo, glossário da cabala); genericamente, trata-se de um catálogo de palavras e o significado que deve ser atribuído a cada uma.

Tesauro: vocabulário de termos relacionados genérica e semanticamente sobre determinada área de conhecimento (TRISTÃO; FACHIN; ALARCON, 2004).

Taxonomia: na Biologia, é o ramo que cuida de descrever, identificar e classificar os seres vivos (classificação sistemática); no estudo das línguas, é a parte da gramática que trata da classificação das palavras; genericamente, trata-se da ciência da classificação.

Ontologia: especificação formal e explícita de uma conceitualização compartilhada (GRUBER, 1993), em que (TRISTÃO; FACHIN; ALARCON, 2004):

- *Conceitualização* significa um modelo abstrato relativo a um conceito relevante relacionado a um fenômeno identificado;
- *Explícito* significa que os conceitos usados e as restrições em relação ao seu uso são definidos explicitamente;
- *Formal* estabelece que a ontologia seja elaborada de tal forma que pode ser lida por uma máquina;
- *Compartilhado* estabelece que ontologia seja capaz de capturar conhecimentos resultantes de consenso, ou seja, aceitos normalmente pelas comunidades envolvidas.

A documentação de recursos, funcionalidades e serviços de forma consistente para evitar redundâncias e ambiguidades pode ser executada utilizando ontologias elaboradas com este fim. Uma ontologia necessita de palavras bem escolhidas para poder descrever os conceitos, atributos, regras e papéis para o seu domínio de conhecimento.

Ontologias se relacionam de forma estreita com os objetos pertencentes ao seu domínio de interesse, sejam eles físicos ou lógicos, bem como suas relações entre eles. São geralmente compostas por Substantivos (objetos) e Verbos (relações) em sentenças que descrevem esse domínio de conhecimento. Enumera os termos importantes e apresenta uma relação de palavras escolhidas, que precisam ser explicadas ou que serão usadas para compor as afirmações. Quais são os termos escolhidos para se abordar? Quais as propriedades que esses termos possuem? O que se deseja falar sobre tais termos? (NOY; MCGUINESS, 2001).

Uma das formas de se estabelecer o escopo de uma ontologia é através de uma lista de questões cuja ontologia deve ser capaz de responder. Essas questões gerais servirão para balizar a abrangência da ontologia e não precisam ser exaustivas: A ontologia contém informações suficientes para responder tais tipos de perguntas? As respostas necessitam um nível particular de detalhamento em alguma área em particular? Isso resultará no estabelecimento da uma competência da ontologia (NOY; MCGUINESS, 2001).

Outro aspecto importante das ontologias é a definição das classes (ou conceitos) e a organização de uma hierarquia entre elas. A elaboração se iniciar pela definição das classes. A partir da relação resultante, os termos que descrevem objetos que possuem existência independente são selecionados (em contraste aos termos que descrevem esses objetos). Esses termos serão âncoras da sua classe. A organização é feita através de uma taxonomia hierárquica. Ou seja, se uma classe A for a superclasse de B, então uma instância de B também será uma instância de A. Isso significa que a classe B representa um conceito que é "do tipo A" (NOY; MCGUINESS, 2001).

As classes sozinhas não dispõem de informação suficiente para responder às questões de competências, conforme visto logo antes. Para isso é preciso descrever a estrutura interna dos conceitos. Exceto os termos usados para indicar as classes, os demais se referem às suas propriedades. Cada classe possui um conjunto característico de propriedades, todas as subclasses herdam suas características. Assim, uma

propriedade deve ser atribuída inicialmente à classe mais genérica possível dentre aquelas que podem possuí-la (NOY; MCGUINESS, 2001).

Dicionário: A palavra "dicionário" tem significado de vocabulário ("*wordbook*") para leitores humanos, segundo dizem alguns autores. A palavra "léxico" é geralmente usada para se referir ao componente de um sistema de PLN que contém informações semânticas (significado) e gramaticais (estruturação e composição) sobre itens lexicais. Uma "base de dados lexical" é aquela que coleciona informações lexicais, com estruturada acessível por sistemas de PLN (GUTHRIE *et al.*, 1996).

Sugere Scapini (1995) que, de maneira geral, "léxico" é uma lista de palavras contendo também seus significados e suas classificações gramaticais. Geralmente está relacionado a um determinado contexto mais ou menos abrangente. Para uma dada língua, um léxico é o conjunto de todos os itens lexicais que seus falantes se utilizam, já utilizaram ou poderão utilizar.

Então, os dicionários (ou léxicos) têm como propósito fornecer informações suficientes sobre palavras tais como: etimologia, pronúncia, morfologia, sintaxe entre outras. Como fornecem definição sobre o sentido das palavras, também apresentam conhecimentos não somente sobre a língua em si, mas também sobre as coisas do mundo. (GUTHRIE *et al.*, 1996).

Como forma de classificação, os léxicos podem ser agrupados de variadas maneiras. Wertheimer (1995) sugeriu cinco categorias:

a) convencionais, com verbetes organizados alfabeticamente;
b) analógicos, com verbetes organizados conforme os significados;
c) etimológicos, organizados conforme a origem das palavras;
d) morfológicos, organizados conforme as flexões dos lexemas; e
e) de sinônimos e antônimos, organizados conforme semelhança e/ou oposição dos significados.

Wilks, Slator e Guthrie (1996) sugeriram agrupá-los por seus objetivos, também em número de cinco:

a) dicionários padrão, explicam o significado das palavras;
b) tesauros ("*thesauri*"), indicam relações entre itens lexicais;
c) dicionários bilíngues, indicam a equivalência entre palavras de línguas diferentes;

d) dicionários de estilo, orientam sobre boas regras de uso conforme a gramática; e

e) dicionários de concordância, são basicamente ferramentas escolares.

Ainda segundo os mesmos autores, há ainda os léxicos (ou dicionários) capazes de serem lidos e tratados por máquinas, no contexto do PLN.

Um exemplo significativo é o *WordNet*, uma base de dados lexical legível por máquina e organizada por significado, segundo seus autores (FELLBAUM, 1998). Apresenta palavras agrupadas por substantivos, verbos, adjetivos e advérbios. Os itens lexicais são apresentados com suas definições, seus possíveis significados e as relações com outros itens lexicais. É estruturado pelo conceito de "*synset*" ("*synonyms sets*" ou conjunto de sinônimos), baseado na sinonímia (relação entre sinônimos), e que estabelece o relacionamento básico construtivo deste dicionário. A relação entre os "*synsets*" estabelece uma hierarquia lexical através da hiponímia, ou seja, a relação existente entre um hiperônimo (significado mais genérico) e um hipônimo (significado mais específico). Por exemplo: {tordo, pisco-de-peito-ruivo} --> {pássaro} --> {animal, ser_animado} --> {organismo, forma-de-vida, ser-vivo} (está originalmente em inglês, assim, apresenta-se como {robin, redbreast } --> {bird} --> {animal, animate_being} --> {organism, life_form, living_thing}). Portanto, o *WordNet* pode ser interpretado como uma Ontologia Léxica, ou seja, uma ontologia a respeito do léxico (HUNTER, 2000).

Outro exemplo de léxico passível de processamento computacional é o "*Explanatory Combinatorial Dictionary*" (ECD – Dicionário Combinatório Explanatório) (MEL'CUK; ZHOLKOVSKY, 1992), modelado sob o conceito "*Meaning-Text Model*" (MTM – Modelo Significado-Texto). Associa os significados aos textos descrevendo os recursos lógicos existentes na Linguagem Natural. Apresenta quadro níveis de representação linguística:

a) semântico;

b) sintático;

c) morfológico; e

d) ortográfico ou fonético.

2.2 MAPAS CONCEITUAIS E REDES SEMÂNTICAS

Uma maneira de estruturar a informação é representando o conhecimento na forma de Mapas Conceituais, organizando os conceitos de forma relacional e modular, em classes e subclasses. Os mapas conceituais baseiam-se em Redes Semânticas, que são uma representação visual do conhecimento, na forma de grafos, onde os conceitos situam-se nos nós e as relações entre os conceitos situam-se nos arcos ou ligações (*links*) entre eles (AMORETTI; TAROUCO, 2000).

Os termos “ontologia”, “mapa conceitual” e “taxonomia” apresentam, considerando as diferença naturais de significado das palavras, uma semelhança evidente entre si. Tanto que Terminólogos e Ontólogos se esforçam em pesquisas no campo da conceituação para elaboração de listas de palavras rigorosamente organizadas em classes. Delas, é possível extrair conceitos como herança semântica, os atributos, as propriedades, as restrições, as instâncias ou membros de classes (KASAMA; ZAVAGLIA; ALMEIDA, 2010).

Há muito tempo as Redes Semióticas, Conceituais ou Semânticas (*Semantic Networks*, em inglês) como também são denominadas, são usadas para representar diversos tipos de conhecimentos na Inteligência Artificial (AMORETTI; TAROUCO, 2000).

A definição de mapas conceituais é uma fase anterior e útil para a preparação de ontologias, na visão da Ciência da Computação. Há similaridade entre mapas conceituais e ontologias. Ambos permitem a avaliação do conhecimento e o refinamento pode ser feito ampliando ou reduzindo o conteúdo (KASAMA; ZAVAGLIA; ALMEIDA, 2010).

A principal e mais comum forma de relação observada em ontologias é a do tipo "é-um/é_uma" (em inglês, *is_a*), que representa de maneira formal e objetiva a herança conceitual existente entre uma subclasse e sua classe. Por exemplo, o conceito "microscopia de varredura por sonda" possui semelhanças conceituais que herda de "microscopia eletrônica de varredura", que igualmente recebeu sua definição a partir do conceito de "microscopia eletrônica". Em contraste, o conceito de “microscopia eletrônica de transmissão” tem herança do conceito de "microscopia eletrônica", mas guarda diferenças significativas em relação ao conceito de “microscopia eletrônica de varredura” (KASAMA; ZAVAGLIA; ALMEIDA, 2010). Isso pode ser representado na figura 2.2.1:

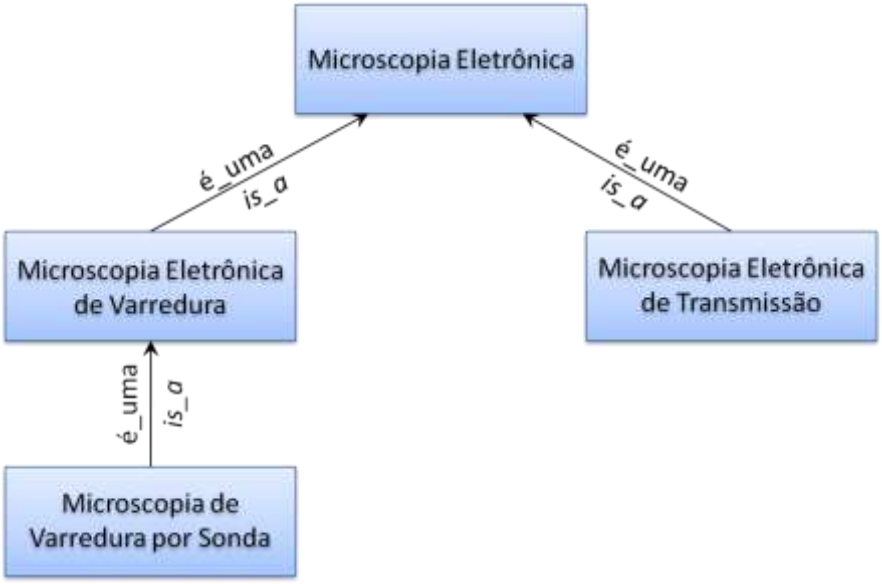


Figura 2.2.1 – Exemplo de relacionamento entre classes e subclasses do tipo "é_uma".
Fonte: (KASAMA; ZAVAGLIA; ALMEIDA, 2010)

Mapas Conceituais são representações gráficas de conceitos e suas relações, podem ser usados como linguagem para descrevê-los ou comunicá-los, podem ser usados tanto para conceitos mais abrangentes quanto para os mais específicos e são estruturados de forma progressiva através do estabelecimento de relacionamento entre os conceitos pré-existentes e os novos, que se pretende inserir. Representam a forma de organização do conhecimento através das relações (as ligações ou *links*) e os conceitos (os nós). O processo de organização do conhecimento permite então avaliar o produto resultante dessas interações (ou os produtos) de forma dinâmica (AMORETTI; TAROUCO, 2000).

Essas ferramentas são também usadas pelas Ciências Cognitivas, como apresentado por Amoretti (2001).

A figura 2.2.2 apresenta um exemplo prático de Mapa Conceitual com conteúdo sobre os próprios Mapas Conceituais.

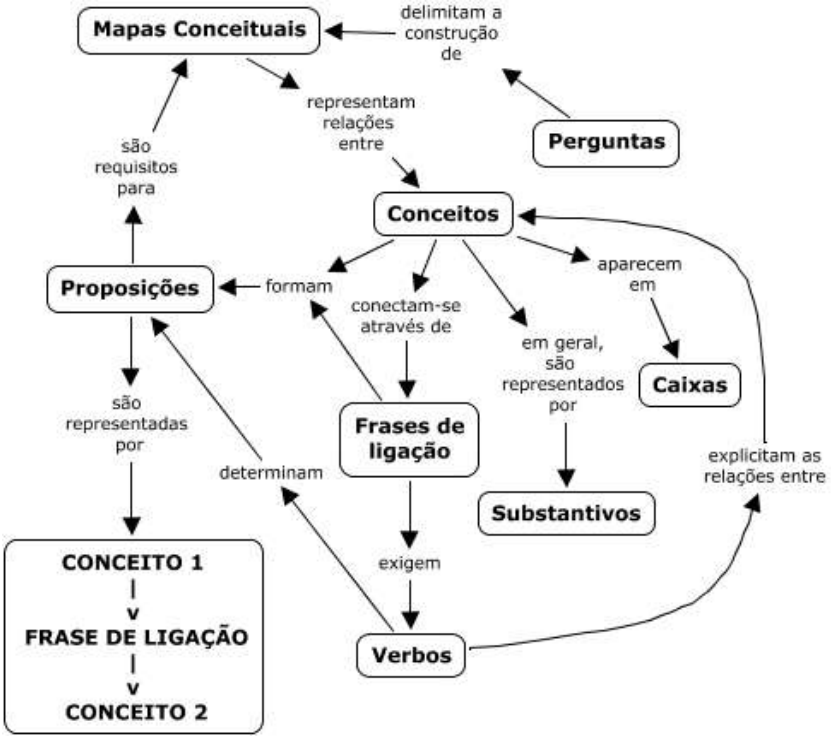


Figura 2.2.2 – Mapa Conceitual sobre Mapas Conceituais.

Fonte: http://lead.cap.ufrgs.br

2.2.1 REPRESENTAÇÃO DO CONHECIMENTO

Os fenômenos linguísticos da polissemia e da sinonímia, respectivamente, denotam que uma mesma palavra pode apresentar mais de um significado ou conceito assim como um significado ou conceito pode ser expresso por mais de uma palavra (PUSTEJOVSKY, 1995) (SCAPINI, 1995) (SACCONI, 1999). Por outro lado, os significados ou conceitos usados para representar as coisas do mundo são únicos por si mesmos e não são ambíguos (CLARK *et al.*, 2000). Assim, a forma lógica de representar o conhecimento é focar nos conceitos. Ontologias adequadas necessitam ser construídas para superar as questões do uso ambíguo das palavras como ocorre na Linguagem Natural. Trata-se de um vocabulário conceitual, sendo que os tesauros (*thesauri*) são usados nessa tarefa (SPARK-JONES, 1986) (LOUKACHEVITCH; SALLI; DOBROV, 1999) (CLARK *et al.*, 2000).

2.2.2 CONHECIMENTO E ONTOLOGIA

Essencialmente, elaborar uma ontologia significa em estruturar seus elementos, ou seja, conceitos e palavras. A forma de organização dessa estrutura pode ser mais facilmente representada por uma árvore, que se denomina taxonomia. Ela apresenta regras para estabelecer como são os relacionamentos válidos. Geralmente se adota a noção de hierarquia objetos com critérios para determinar como os descendentes herdam as características dos ancestrais (JURAFSKY; MARTIN, 2000).

Para representar e referenciar os conceitos de uma ontologia são usadas palavras disponíveis no léxico da língua em que é elaborada (ou línguas) (BOUILLON, 1998). Como os léxicos fornecem definição sobre o sentido das palavras, também apresentam conhecimentos não somente sobre a língua sem sim, mas também sobre as coisas do mundo (GUTHRIE *et al.*, 1996). Os tesauros ("*thesauri"*) são uma maneira de organizar o léxico dessa língua.

*2.2.4 TESAUROS ("*THESAURUS*")*

A bibliografia especializada apresenta diversificada definição para Tesauro ("*thesaurus*"). A maioria reforça que a melhor prática está no foco nos conceitos, e não nas palavras, para referenciar a base de relacionamento do léxico disponível (GONZALEZ, 2001).

O objetivo do tesauro é permitir que, a partir de qualquer conceito identificado por um item lexical, seja possível identificar os demais itens lexicais relacionados e o grafo usado na construção deve ser dirigido para atender a esse objetivo. A relação mais comumente encontrada em Tesauros a de sinônimos, mas é possível elaborar também relações de antônimos (significados contrários), significados específicos, abrangentes ou genéricos, partes integrantes ou constituintes, derivação de significados etc.

2.2.5 RELAÇÃO COM A PROPOSTA DESTE TRABALHO

Para que os resultados almejados sejam alcançados, tanto no planejamento quanto na execução, é preciso estabelecer conceituações claras sobre os recursos técnicos oferecidos pelos dispositivos escolhidos para a implementação de um sistema bem como os serviços que se pretende oferecer. Para isso, as ontologias se mostram como ferramentas produtivas para realizar a documentação ou o registro.

2.3 INTELIGÊNCIA ARTIFICIAL E APRENDIZADO DE MÁQUINA

2.3.1 DADO, INFORMAÇÃO, CONHECIMENTO, INTELIGÊNCIA E SABEDORIA

Segundo Rowley (2006), uma ordem natural pode ser observada entre os conceitos dado, informação e conhecimento. É neste contexto que Ackoff (1989 *Apud* ROWLEY, 2006), autor frequentemente citado na literatura recente, oferece as seguintes definições:

- Dado é definido como símbolos que representam propriedades de objetos, eventos e seus ambientes. Resultam da observação. Não apresentam utilidade até que sejam tomados de forma aplicável (relevância). A diferença entre Dado e Informação é funcional e não estrutural;
- Informação está contida em descrições, respostas a questões iniciadas com palavras tais como Quem, O que, Quando e Quantos(as). Sistemas de informação geram, armazenam, recuperam e processam dados. A informação é inferida a partir de Dados.
- Conhecimento é saber como e é o que torna possível transformar informação em instruções. O conhecimento pode ser obtido por transmissão de alguém que o possui, por instruções, como extraindo-o da experiência.
- Inteligência é a habilidade de aumentar a eficiência.
- Sabedoria é a habilidade de aumentar a eficácia. A sabedoria agrega valor, o que requer a função mental denominada Julgamento. Os valores éticos e estéticos aqui implicados são inerentes do ator e são únicos e pessoais.

Discutindo a proposição de Ackoff, Zeleny (1987 *Apud* ROWLEY, 2006) e Bellinger *et al*. (2004 *Apud* ROWLEY, 2006) fizeram proposições úteis bem como modificações na hierarquia, elaborando um pouco mais o exposto por Ackoff.

Segundo Rezende (2003), a figura 2.3.1 representa graficamente as ideias contidas nos três modelos (Ackoff, Zeleny e Bellinger) para o relacionamento entre esses conceitos.

Figura 2.3.1 – Dado, Informação, Conhecimento e Inteligência.

Fonte: (REZENDE, 2003)

2.3.2 RELAÇÃO COM A PROPOSTA DESTE TRABALHO

A hierarquia dado-informação-conhecimento-sabedoria (sigla DIKW do inglês "*data–information–knowledge–wisdom*") é um dos modelos mais reconhecidos nas literaturas sobre gerenciamento da informação, sistemas de informação e gerenciamento do conhecimento, segundo Rowley (2006). É usada para contextualizar dado, informação, conhecimento e, às vezes, inteligência e sabedoria, um em relação ao outro e para descrever o processamento que transforma uma entidade de um nível em outra do nível seguinte (por exemplo, dados em informação).

Esta maneira de pensar, ou seja, agrupar os conceitos semelhantes e organizá-lo em uma ordem, que pode ser de significação, importância ou entendimento, facilita a compreensão e a aplicação dos conceitos expostos ao longo deste trabalho.

A seguir serão apresentadas algumas técnicas computacionais conhecidas como Aprendizado de Máquina.

2.4 MODELO DE ENTROPIA MÁXIMA

Para o processamento da linguagem natural, para que o resultado da análise seja obtido, é preciso executar algumas tarefas no texto tais como: *separação das sentenças*, *separação das palavras*, *classificação das palavras segundo a função sintática* e *interpretação do significado*.

Do ponto de vista computacional, tais tarefas podem ser executadas através de diversas técnicas, que podem ser agrupadas em dois grupos: *métodos determinísticos* e *métodos estocásticos* ou estatísticos. Os métodos determinísticos se baseiam em regras que determinam a formação do texto para a análise. O Capítulo 3 aborda os métodos determinísticos utilizados mais comumente. Os métodos estocásticos, por outro lado, não se preocupam tanto com as regras, mas em extrair os padrões a partir de ocorrências reais por meio da estatística.

Um classificador é uma ferramenta de aprendizado de máquina capaz de separar itens de dados em classes pré-determinadas. Um classificador probabilístico pode informar a distribuição de probabilidade da classe designada atribuída para o item. (MANNING; KLEIN, 2003). Ao caracterizar eventos desconhecidos com um modelo estatístico, deve-se escolher aquele que apresente a máxima entropia (ZHANG, 2007). A Entropia Máxima é uma técnica de uso geral de aprendizado de máquina que proporciona a estimativa com menor tendência possível baseada em uma dada informação (VACHASPATI; WU, 2012).

Segundo Zadrozny (2008), trata-se de um modelo computacional estatístico utilizado para conhecer a probabilidade do conteúdo de uma variável pertencer ou não a uma classe, de um conjunto finito de classes, baseando em critérios múltiplos e que não prevê quaisquer restrições ou regras quanto às sequências possíveis para os dados dessa variável. Ou seja, definido um conjunto classes e conjunto de dados que podem se apresentar em qualquer sequência sem restrições, o modelo de entropia máxima é usado para retornar a probabilidade de um dado pertencer ou não a uma dessas classes, considerando o resultado de classificações observadas anteriormente e que foram executadas considerando múltiplos critérios de classificação.

2.4.1 ENTROPIA MÁXIMA

Assumindo que Entropia seja a medida do caos ou desordem de um sistema, Entropia Máxima refere-se ao limite da maior desordem frente certas condições desejadas. Então, sob certas circunstâncias, ao se aplicar os princípios da máxima entropia em uma situação que se deseja estudar resultará num modelo bastante genérico desse problema (ENGELS, 2011).

A forma de abordagem proposta pelo Modelo de Entropia Máxima é estabelecer um mecanismo capaz de ser "treinado" e que exiba seus resultados na mesma proporção em que são encontrados nos dados de treinamento, ao contrário de se criar um mecanismo que simplesmente gere os resultados encontrados nos dados de treinamento. No caso, exemplos de resultados seriam as classes gramaticais das palavras, a separação entre uma palavra e outra, a identificação de nomes, de números, de datas etc. (ENGELS, 2011).

Então, o que delimita as circunstâncias mencionadas antes, são esses aspectos desejados sobre a sentença e a entropia máxima retratará o modelo mais genérico possível para esse conjunto conhecido de aspectos delimitadores. O treinamento é realizado com relação a esses aspectos e não em relação às palavras em si devido ao seu maior valor (maior interesse). Também, quando os dados de treinamento são esparsos, é difícil obter modelos bem treinados. Por outro lado, um conjunto reduzido de resultados esperados pode ser mais facilmente obtido a partir de dados esparsos ou mesmo a partir de uma grande porção de dados de treinamento (ENGELS, 2011).

Conforme esclarece Zadrozny (2008), os Modelos de Entropia Máxima, também denominados como *MaxEnt* na literatura, se propõem a minimizar os erros através da máxima entropia ou exposição a todas as combinações possíveis de resultados encontrados na realidade, modelando tudo o que é conhecido e não presumindo nada a respeito do desconhecido. Segundo a autora, o tipo de análise de regressa utilizado pelo *MaxEnt* se ajusta melhor à natureza dos dados encontrados no processamento da linguagem natural, que não se apresentam com uma distribuição normal.

2.4.2 CONSTRUÇÃO DE UM MODELO DE ENTROPIA MÁXIMA

Dispondo de dados para realizar o treinamento em volume suficientemente grande e diversificado de resultados corretos (dados previamente classificados) e do conjunto de classes, inicia-se o modelo com os parâmetros iniciais ajustadas para valores arbitrários.

Os parâmetros do modelo são reestimados usando a frequência de ocorrência das palavras e classes como observados nos dados de treinamento.

Uma comparação é então feita no resultado das probabilidades obtidas a partir do modelo, comparando com o observado nos dados de treinamento. Havendo diferença nos valores, os parâmetros são reestimados novamente.

O processo de computação é repetido ciclicamente até que um limite aceitável de erro ou um número máximo de iterações seja atingido (ENGELS, 2011).

2.5 AUTÔMATO DE ESTADOS FINITOS

2.5.1 CONCEITUAÇÃO

Segundo Black (2008) a teoria dos autômatos é um ramo teórico da ciência da computação e surgiu no início do século 20 quando matemáticos iniciaram o desenvolvimento teórico e real de máquinas capazes de imitar certas habilidades humanas, executando cálculos mais rápida e eficientemente. Semelhante à palavra automação, denota “processos automáticos conduzindo a produção de processos específicos”.

Na computação, é parte do que se denomina “programação dinâmica”, que propõe a solução de problemas complexos através da sua divisão em partes menores.

Autômatos são abstrações de máquinas que processam uma entrada mediante uma série de transições de estados ou configurações. Auxiliam cientistas a compreender como uma máquina pode computar uma função e resolver problemas, facilitando meios de desenvolver métodos capazes de descrever e analisar o comportamento de sistemas discretos. Portanto, as características básicas de máquinas assim incluem:

- Entradas: sequência de símbolos de um conjunto finito de elementos;
- Saídas: sequência de símbolos de um conjunto finito de elementos;
- Estados: conjunto finito de estados característicos do autômato.

O primeiro autômato finito foi descrito por dois neurofisiologistas Warren McCulloch e Walter Pitts em 1943 no artigo "*A Logical Calculus Immanent in Nervous Activity*". Mais tarde, dois outros cientistas da computação Mealy e Moore generalizaram a teoria em máquinas mais poderosas.

Uma Máquina de Estados Finitos, ou Autômato de Estados Finitos, é aquela cujo conjunto de estados possui quantidade finita de elementos. Seu funcionamento parte de um estado iniciar e uma entrada, em seguida apresenta uma nova configuração na saída e o estado seguinte. Apresentam pequena memória e não mantém registro dos estados anteriores.

Formalmente é definido por uma quíntupla (*Q, I, Z, Y, W*), onde:

Q = conjunto finito de estados;

I = conjunto finito de entradas;

Z = conjunto finito de saídas;

Y = função de transição de estado $I \times Q \rightarrow Q$;

W = função de saída $I \times Q \rightarrow Z$.

Exemplo para Autômato de Estados Finitos seria um elevador, mecanismo que não registra os andares servidos anteriormente, mantém registro do andar atual, se o movimento é para cima ou para baixo e as chamadas ainda a serem atendidas.

2.5.2 RELAÇÃO COM A PROPOSTA DESTE TRABALHO

A teoria dos autômatos é utilizada geralmente como forma de executar o processamento da linguagem natural, para implementação das regras gramaticais, por exemplo, permitindo que certa sequência correta de palavras de uma frase seja aceita como válida e, assim, sinalizar o reconhecimento de um padrão possível. Avaliando o conjunto de sinalizações dessa cadeia é possível extrair alguma conclusão, realizar uma transformação ou extração de significado, por exemplo.

2.6 CADEIA OCULTA DE MARKOV

Aplicadas com sucesso em diferentes formas de processamento de texto, os modelos de Cadeia Oculta de Markov (em inglês, *Hidden Markov Model* ou HMM) são ferramentas poderosas o suficiente para modelar sequências de dados. Em aplicações como rotulagem de classes gramaticais, segmentação de texto e extração de informação,

a observação é normalmente modelada frequentemente como distribuição multimodal de um vocabulário discreto. Nestes casos, os parâmetros do HMM são ajustados para maximizar a probabilidade das observações (MCCALLUM; FREITAG; PEREIRA, 2000).

Um Modelo de Cadeia Oculta de Markov (em inglês, *Hidden Markov Model* ou HMM) é um autômato de estados finitos, que apresenta a transição de estados e sua observação de maneira estocástica. Esse autômato modela uma sequência de observações por meio de um processo gerativo probabilístico. Este processo se inicia num estado qualquer e emite uma observação que é determinada por este estado. Em seguida, há uma transição de estado e a emissão de uma nova observação. E o processo segue até que um determinado estado final é alcançado (MCCALLUM; FREITAG; PEREIRA, 2000).

Em termos mais formais, o HMM é formado por:

S conjunto finito de estados;

O conjunto de observações possíveis;

$P(s|s')$ distribuição de probabilidade de transição do estado s' para s, ambos pertencentes a S;

$P(o|s)$ distribuição da probabilidade de observação;

$P_0(s)$ distribuição de probabilidade no estado inicial.

Durante a execução será obtida a sequência de observações $o_1, o_2, \ldots o_m$ respectivamente para os estados $s_1, s_2, \ldots s_m$.

Em se tratando de texto, essas observações são na realidade um determinado conjunto de letras ou um determinado vocabulário.

Em se tratando de extração de informação, estará em questão uma sequência de etiquetas ou rótulos (em inglês, *labels*) $l_1, l_2, \ldots l_m$ relativa a cada observação de treinamento $o_1, o_2, \ldots o_m$. Normalmente nesses modelos, são associados um ou mais estados a cada etiqueta possível. E para uma cada nova observação, o objetivo é encontrar a sequência de etiquetas mais provável.

2.7 MODELO DE MARKOV DE ENTROPIA MÁXIMA

Os Modelos de Markov de Entropia Máxima ("*Maximum Entropy Markov Model*", ou MEMM em inglês) combinam os recursos da Cadeia Oculta de Markov com o Modelo de Entropia Máxima. Ou seja, referem-se a uma técnica que busca definir um

modelo a partir de resultados observados, porém pressupondo a existência de restrições ou regras que definem a ocorrência na sequência dos dados (Zodrozny, 2008).

Consiste de uma forma de ajustar uma sequência de elementos (no caso, palavras de uma sentença) ao conjunto mais provável de transições de uma aplicação de autômato de estados finitos que é a Cadeia de *Markov*. Supondo que cada estado corresponda a um estágio conceitual da sentença e cada estado pode emitir certas palavras, o caminho mais provável através dessa máquina de estados seria o que melhor caracterizaria a sequência de palavras (ENGELS, 2011).

Uma das vantagens desta solução é que permite a especificação de correlações úteis entre os dados. Também, o esforço computacional para treinamento dos modelos resulta mais eficiente (MCCALLUM; FREITAG; PEREIRA, 2000).

2.8 ALGORITMO DE BUSCA DO MELHOR CAMINHO

Denomina-se algoritmo de resolução de problema por meio de busca aquele que tem em sua entrar um problema e retorna na sua saída uma sequência de ações a serem tomadas. Essa sequência de ações levará à solução do problema quando executadas (RUSSEL; RUSSEL, 2004). Trata-se de técnica de Inteligência Artificial.

Genericamente, segundo os mesmos autores, é possível dizer que um agente capaz de resolver um problema precisa executar os passos “formular, buscar e executar”. Em outras palavras, depois de formulado um problema a resolver (e estabelecer seu objetivo), um procedimento de busca é executado e seu resultado estabelece o que deve ser executado para resolver o problema, atingindo assim o objetivo.

Um algoritmo de busca sem informação é aquele que procura a solução somente com a definição do problema e sem nenhuma informação de onde buscar as respostas. Já um algoritmo de busca com informação, também denominado de busca heurística, é aquele que procura a solução dispondo de informações de onde a resposta pode ser encontrada (RUSSEL; RUSSEL, 2004).

Encontrar uma rota, ou seja, a determinação prévia do caminho necessário para se atingir um determinado destino a partir de um dado ponto de origem, é um dos exemplos didáticos sobre algoritmos de busca. O problema do planejamento de rota surge quando o caminho entre a origem e o destino não pode ser uma reta ou outra curva que se possa repetir sempre, ou seja, há obstáculos a serem desviados e o ponto de

origem ou de destino, ou ambos, podem ser diferentes entre dois momentos em que o problema surge.

Por razões de desempenho, um aspecto desejável do planejamento de rota é que o algoritmo não deve vasculhar todo o ambiente em busca da solução, mas a menor área possível, para chegar até a resposta de forma objetiva.

Entre os algoritmos de busca que foram pesquisados, os seguintes se apresentaram como mais promissores para aplicação no sistema robótico proposto:

- Dijkstra (MORRIS, 1998);
- A* ou *A Star* em inglês ou A-Estrela (HART; NILSSON; RAPHAEL, 1968);
- D* ou *D Star* em inglês ou D-Estrela (STENTZ, 1994); e
- *D* Lite* (KOENIG; LIKHACHEV, 2002).

Estes algoritmos também são encontrados em variações e versões modificadas, otimizados para alguma aplicação especial. Foram avaliadas somente as versões originais.

O algoritmo *Dijkstra* é indicado como o primeiro deles, sendo, portanto, o mais antigo. Foi concebido em 1956 e publicado em 1959. É um algoritmo de força bruta, ou seja, avalia **todas** as possibilidades e escolhe a que consiste no menor percurso. Para problemas com número pequeno de pontos a serem considerados e de possíveis soluções (como, por exemplo, encontrar a melhor rota entre duas cidades), ele pode ser aceitável. Porém, para situações reais e mais complexas como encontrar o melhor trajeto numa planta com várias divisões, corredores e salas, o custo computacional pode ser bastante elevado. Assim, optou-se por não utilizar este algoritmo.

O algoritmo A* (A-Estrela), ou *A Star* em inglês, é um dos mais antigos. Trata-se de uma extensão do algoritmo *Dijkstra* e foi primeiramente descrito em 1968. Apresenta um ganho computacional, resultando em melhor desempenho, uma vez que não requer a computação de todos os caminhos possíveis, sendo capaz de avaliar o custo até o próximo ponto de cada passo do movimento, não só o custo total até o destino. Com isso, é possível limitar a quantidade de alternativas a serem computadas. Uma vez que é algoritmo de entendimento relativamente fácil, é relativamente simples de implementação computacional e possui algumas versões de implementações (desde a mais didática até a mais eficiente computacionalmente) disponíveis na comunidade de *software* livre, foi o algoritmo escolhido para aplicação no sistema robótico proposto.

O algoritmo D*, ou *D Star* em inglês, é também conhecido como "*Dynamic A**" (ou *A Star* Dinâmico) foi introduzido em 1994. É um algoritmo mais utilizado em robótica, pois diferentemente do A* que pressupõe a existência de ambientes estáticos (que não apresentam mudanças ao longo do tempo), permite melhor eficiência computacional quando utilizado em ambientes dinâmicos, ou seja, os obstáculos do ambiente são móveis. Este é um inconveniente do algoritmo A*, que executa o recálculo completo da rota quando se depara com uma mudança na planta, ao passo que o D* reaproveita os cálculos anteriores alterando somente o necessário após a constatação da mudança. Optou-se pela não adoção deste algoritmo graças à existência do algoritmo *D* Lite*, comentado no próximo parágrafo.

O algoritmo *D* Lite* foi introduzido em 2002. Apesar do que o nome parece sugerir, este não se trata de uma evolução do D*, mas sim de uma evolução a partir de uma variação do A* denominada LPA* (*Lifelong Planning A** em inglês*)* e que apresenta o mesmo comportamento de uma versão modificada daquele, denominada *Focused D** (ou D* Focalizado). O funcionamento do *D* Lite* é mais fácil de ser compreendido do que as variações do D*, sendo também computacionalmente mais fácil de ser implementado (a palavra "*lite*" remete à expressão "leve"). Porém, por ser algoritmo mais recente e por não ter sido possível encontrar estudos de casos de sua implementação em C#, sua adoção não foi possível. Para tanto, a quantidade de horas de trabalho necessárias para estudo, desenvolvimento, implementação e testes facilmente poderia esgotar o prazo. Assim, uma recomendação de desenvolvimento futuro é o estudo da aplicação deste algoritmo, principalmente considerando que o DRDC (*Defense Research and Development Canada)* o considerou como solução tecnológica promissora (MACKAY, 2005).

O algoritmo A*:

O algoritmo de planejamento de rota mais conhecido é a Busca A* ("*A Star*" em inglês, ou A-Estrela). Trata-se de uma busca heurística que compara o custo total estimado da solução e retorna aquela com o menor custo (RUSSEL; RUSSEL, 2004).

Para efeito prático, a quantidade genérica "custo" de uma rota pode ser associada a uma grandeza mensurável como distância ou tempo, por exemplo, e que faça sentido para determinar a melhor rota dentro do contexto do problema em questão.

O custo total de uma rota é valor do custo para ir da origem até um ponto qualquer da rota somado ao custo para seguir adiante, a partir deste mesmo ponto até o destino. Matematicamente, a representação pode ser dada por:

$$f(n) = g(n) + h(n) \qquad \text{(eq. 2.8.1)}$$

Ao considerar que g(n) fornece o custo real da rota desde a origem até o ponto *n* e h(n) fornece o custo estimado desde o ponto *n* até o destino, passando pelo caminho mais econômico, é possível dizer que:

f(n) = menor custo estimado para a solução que passa pelo ponto *n*.

A forma mais direta de fazer uma estimativa do restante do caminho – h(n) – é uma linha reta, em se tratando de espaço euclidiano. Trata-se de uma solução otimista, pois o custo real da solução será provavelmente maior que a estimativa em linha reta. Mesmo assim, com essa estimativa otimista Hart, Nilsson e Raphael (1968) demonstram que a Busca A* é capaz de encontrar uma solução que é ao mesmo tempo completa e ótima.

Porém, a estimativa denominada Euclidiana, do ponto de vista computacional, apresenta um inconveniente. Ela requer cálculos de ponto flutuante com raiz quadrada e exponenciação (quadrado), que pesam no desempenho geral e o que se busca pode não justificar tanto sacrifício em nome da precisão, uma vez se tratando de determinar uma estimativa. Uma representação da distância Euclidiana encontra-se na figura 2.8.1.

Assim, um melhor desempenho computacional pode ser obtido usando aritmética de números inteiros. Na estimativa denominada *Manhattan* o custo restante para o caminho é feita de forma semelhante ao caminhar a pé pelas ruas, através de quarteirões e esquinas. Como não é possível caminhar através das quadras, mas somente pelas calçadas e ruas, as distâncias são descritas por dois eixos perpendiculares entre si, sendo que a distância total será a soma numérica simples da distância percorrida em cada eixo. Uma representação da distância *Manhattan* encontra-se na figura 2.8.2.

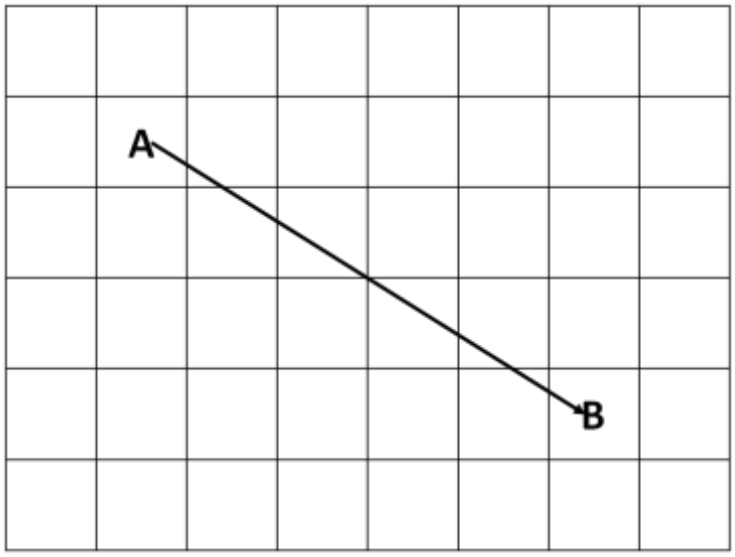


Figura 2.8.1 – Distância "Euclidiana" entre dois pontos.

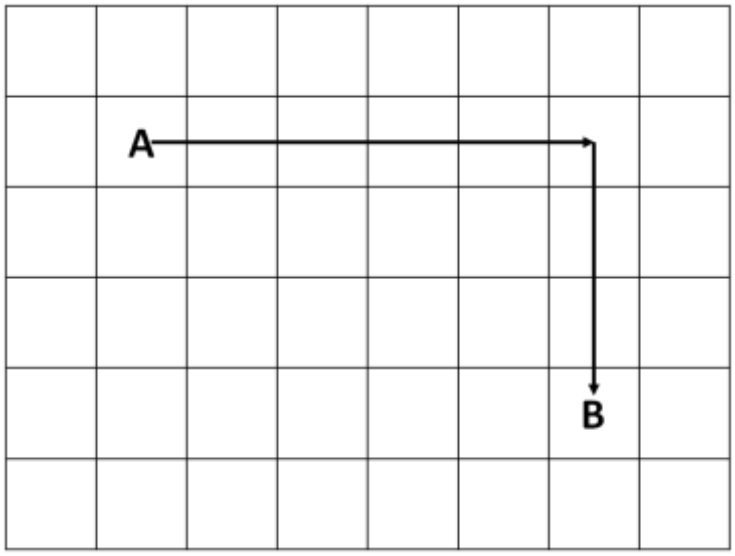


Figura 2.8.2 – Distância "*Manhattan*" entre dois pontos.

Tentando minimizar o erro da distância adicional decorrente dessa estimativa simplista, foi proposta por Seder, Petrovi e Maček (2008) outra forma de estimar a distância e que será utilizada no sistema proposto por este trabalho. É possível calcular essa estimativa com as equações 2.8.2:

$$a = máximo(|xT - xn|, |yT - yn|)$$
$$b = mínimo(|xT - xn|, |yT - yn|)$$
$$h(n) = 14b + 10(a - b) \quad \text{(eq. 2.8.2)}$$

Onde as coordenadas do ponto de destino, ou ponto final, *T* são (xT, yT) e as coordenadas do ponto atual sob análise *n* são (xn, yn). Segundo os autores, esse método exibe monotonicidade em decorrência da propriedade da desigualdade triangular.

Quando se trata do planejamento de rotas, um mapa ou uma planta apresentam informações suficientes para se buscar a solução de um problema do tipo "partir do ponto inicial e seguir o melhor caminho possível até o ponto final".

Com isso, o algoritmo da Busca A* pode então ser descrito da seguinte forma (HART; NILSSON; RAPHAEL, 1968):

1) Adicionar o ponto inicial *s* na "lista aberta" e calcular f(s);

2) Avaliando todos os pontos *n* adjacentes ao ponto de partida que sejam rotas possíveis (não sejam obstáculos) em direção ao ponto de destino, calcular o valor f(n) para cada um deles, vincular o ponto atual como antecessor de cada um desses novos pontos, acrescentá-los na "lista aberta" e escolher o novo ponto que tiver o menor valor para f(n);

3) Se o novo ponto *n* for o ponto de destino *T*, então ele deve ser acrescentado na "lista fechada" e o algoritmo terminado;

4) Caso contrário, remover o ponto atual da "lista aberta", acrescentá-lo na "lista fechada" e aplicar o operador de sucessão deslocando as referências para o novo ponto escolhido. Avaliando todos os novos pontos adjacentes a *n* que sejam rotas possíveis, calcular o valor f(n) de cada um e todos que não estiverem na "lista fechada" devem ser acrescentados na "lista aberta", vinculando o ponto atual como o antecessor deles. Os pontos que estiverem na "lista fechada", e que tenham agora, com a nova referência, um valor f(n) menor que a estimativa da verificação anterior, devem ser acrescentados na "lista aberta" considerando-os vinculados ao ponto atual como o novo antecessor. Verificar e fazer o mesmo para os pontos da "lista aberta". Ir para o passo 2.

A "lista aberta" contém os pontos mais interessantes, ou seja, aqueles que provavelmente fazem parte do caminho e precisarão ser verificados outras vezes.

A “lista fechada” é mantém os pontos menos interessantes, ou seja, aqueles que provavelmente não fazem parte do caminho e não precisarão ser verificados por enquanto.

Uma vez atingido o ponto de destino, a rota resultante da Busca A* pode ser encontrada seguindo os pontos um a um pela vinculação de antecessor, no sentido contrário, ou seja, desde o ponto de destino *T* até o de origem *s*.

Com um exemplo será possível compreender melhor como funciona o algoritmo.

2.8.1 INICIANDO A BUSCA

Dada uma planta qualquer, se deseja deslocar a partir do ponto inicial, que é a posição atual, até o ponto final, ou de destino. Como contextualizado anteriormente, um caminho reto de um ponto ao outro não é possível devido à presença de obstáculos. Então, será preciso encontrar a melhor rota entre esses pontos. A figura a 2.8.3 é um exemplo de uma situação como essa.

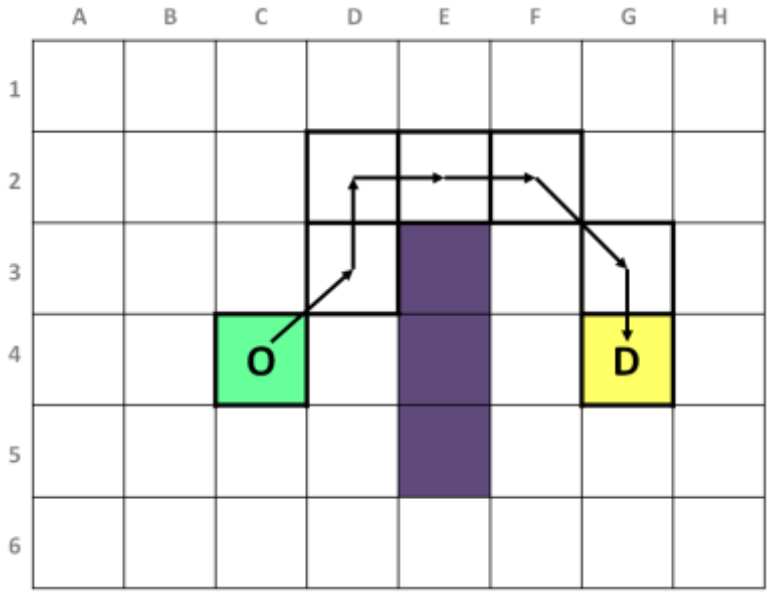


Figura 2.8.3 – Exemplo de mapa com pontos de origem e destino, obstáculo e uma rota.

O algoritmo A* encontra a solução do problema examinando os pontos adjacentes objetivando determinar qual o melhor ponto para realizar o próximo passo

do movimento. Isso é repetido em ondas sucessivas até que o ponto final seja encontrado, caso exista uma rota possível. Em cada onda de execução, o controle é feito colocando os pontos encontrados ou na "lista aberta" e ou "lista fechada".

Quando um ponto é acrescentado em uma dessas listas, é feita uma vinculação de qual foi o seu ponto antecessor, ou seja, em qual sentido e direção foi o passo dado, ou ainda, qual o vetor do deslocamento.

Outras informações, vinculadas aos pontos quando forem acrescentados em uma das listas, são relativas ao custo de deslocamento, tanto para se chegar até o ponto, quanto para prosseguir a partir dele até o destino. Sabendo que o custo pode ser avaliado pela equação 2.6.1, cada ponto presente em cada lista será representado pelos seguintes parâmetros: f(n), g(n), h(n) e vetor de deslocamento (a partir do antecessor).

Partindo do ponto inicial, avaliando os pontos adjacentes em todas as direções possíveis, calculando os parâmetros e indicando o vetor de deslocamento, o resultado pode ser representado pela figura 2.8.4.

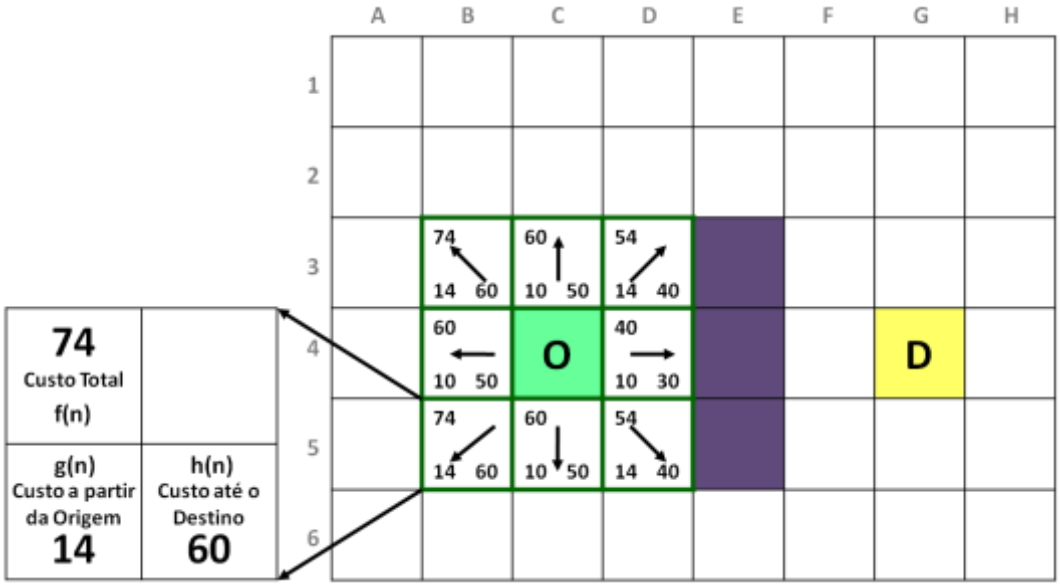


Figura 2.8.4 – Arredores da origem avaliados, antes de fazer o primeiro passo.

Para facilitar, será considerado o centro do quadrado como ponto de referência, tanto para indicar o vetor de deslocamento quanto para cálculo de distância/custo.

Uma generalização foi feita assumindo que o custo de deslocamento é linearmente proporcional à distância, qualquer que seja a direção. Mas pode ser usado custo diferente, se isso for útil na resolução do problema. Exemplos disso são: a presença de aclives, uma mudança de tipo de piso ou outro fator que seja útil sinalizar como penalização de custo para o movimento.

O deslocamento pode acontecer em oito direções, ou seja, na horizontal, na vertical e nas diagonais. Porém, não é impossível que aconteçam casos em que se queira restringir o movimento somente nas perpendiculares.

Considerar que o deslocamento nas perpendiculares possui custo 10. Para o deslocamento nas diagonais será um pouco maior já que a distância entre os centros dos quadrados também o é. Aplicando o Teorema de Pitágoras, obtém-se que o valor da raiz quadrada de 2 é aproximadamente 1,41 ($\sqrt{2} \cong 1{,}41$). Portanto, utilizando números inteiros, é possível aproximar para 14 o custo do movimento diagonal.

Em benefício da simplicidade, neste exemplo, a estimativa da distância até o destino será feita pelo método "*Manhattan*".

Para prosseguir com a execução do algoritmo A*, o passo 2 indica simplesmente para escolher o ponto com menor valor de f(n). Neste caso, será escolhido o ponto D4.

2.8.2 CALCULANDO OS VALORES PELA PRIMEIRA VEZ

Ao escolher o novo ponto e aplicar o operador de sucessão, o ponto anterior (no caso, o ponto de origem) é removido da "lista aberta" para a "lista fechada" e o ponto D4 passa ser a referência para a execução do passo 3 do algoritmo.

Avaliando os pontos adjacentes, é possível observar que nenhuma das novas estimativas para f(n) se alteram, pois as anteriores já resultaram nos valores mais baixos.

E o algoritmo recomenda simplesmente escolher, então, o ponto com menor valor de f(n). Os pontos constantes da "lista fechada" não são interessantes. Os pontos que fazem parte dos obstáculos não fazem parte da nenhuma rota possível, pois são intransponíveis. Os pontos restantes já se encontram na "lista aberta" e há duas opções com o menor valor para f(n). Não faz diferença significativa qual das duas escolher neste caso. Do ponto de vista da execução pelo computador, é mais simples usar o primeiro

valor da “lista aberta” e economizar alguns ciclos de relógio. Neste caso, supõe-se que seja o ponto superior, ou seja, D3.

2.8.3 PROSSEGUINDO COM A BUSCA

Agora, avaliando os pontos adjacentes, o ponto bem a direita pertence ao obstáculo e deve ser ignorado, assim como o que está abaixo dele. O ponto bem acima da barreira, o E2, pode ser ignorado ou não, dependendo da regra estabelecida para movimentação. Caso não haja distância suficiente para esse deslocamento diagonal pela presença da quina da barreira, o robô esbarraria ou seria bloqueado no seu movimento. No caso em questão, o movimento diagonal adjacente a um obstáculo não é permitido e o ponto E2 será ignorado nesse momento.

Dos cinco outros pontos restantes, dois já pertencem à “lista fechada” (o ponto de origem e o imediatamente anterior) e podem ser desconsiderados. Acima estão os pontos C2 e D2 que ainda não foram vistos e podem ser incluídos na “lista aberta”, vinculando o ponto atual como seus antecessores e calculando g(n), h(n) e f(n) para eles. Para o último ponto restante, imediatamente à esquerda (C3), deverá ser verificado se um movimento até ele resultaria alguma vantagem, ou seja, um valor de g(n) menor. Neste caso não e deverá ser mantido o valor atual. A situação neste ponto da execução pode ser visualizada pela figura 2.8.5.

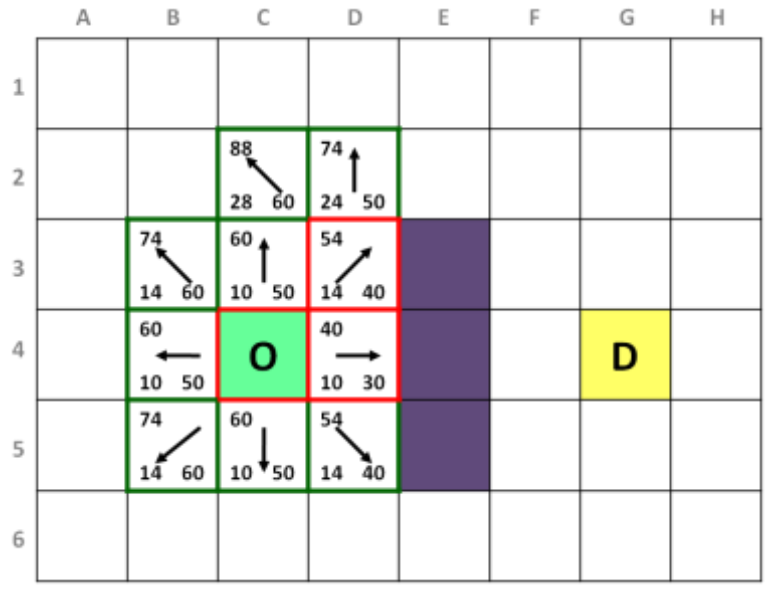


Figura 2.8.5 – Arredores do ponto D3 avaliados, antes de fazer o próximo passo.

Seguindo o algoritmo e escolhendo o ponto com menor valor para f(n), o passo seguinte será para o ponto C3. Aplica-se o operador de sucessão, o ponto anterior (D3) é removido da "lista aberta" para a "lista fechada" e o ponto C3 será a referência para a execução do passo 3 do algoritmo.

Apesar de este ser um passo sem muita utilidade, o funcionamento normal do algoritmo explora as possibilidades de movimento em busca da melhor solução. Mesmo assim, o resultado no final será a melhor rota entre a origem e o destino. Também, a varredura dos pontos para determinar esse caminho será minimizada significativamente. Diferentes formas de estimar h(n) resultarão em sequências diferentes de exploração, dependendo do formato e da disposição das barreiras.

Com relação ao ponto C3, há oito pontos a serem explorados. Três deles estão na "lista fechada" e serão desconsiderados: D4, D3 e a origem. Prosseguindo pelos pontos que constam da "lista aberta" e avaliando g(n) para eles, verifica-se que não há nenhum ganho e seus valores continuam sem alteração.

Há um ponto que ainda não foi vista antes, B2, que é incluído na "lista aberta", o ponto atual (C3) é registrado como seu antecessor e os valores de g(n), h(n) e f(n) são calculados. É interessante observar que para um dos pontos constantes da "lista aberta", há um ganho em recalcular g(n), o custo a partir da origem: C2.

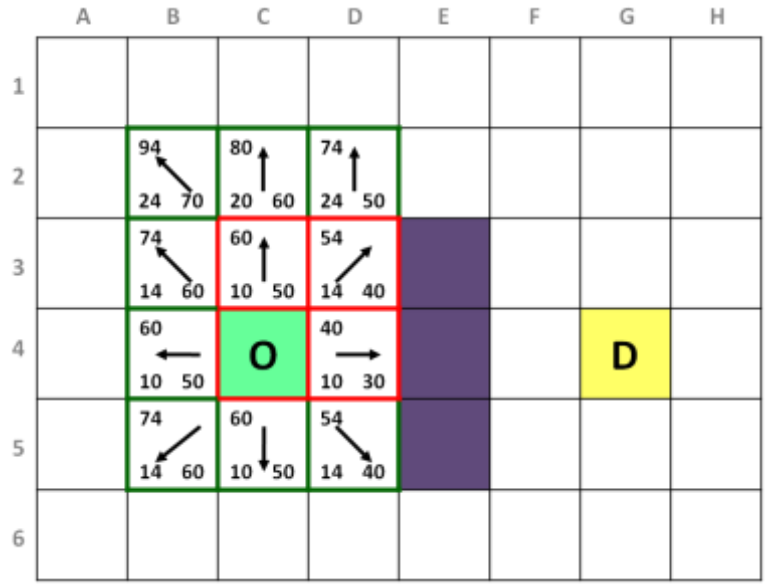


Figura 2.8.6 – Arredores do ponto C3 avaliados, antes de fazer o próximo passo.

Terminando de verificar os arredores e antes de seguir para o próximo passo, que seria escolher o ponto com menor valor para f(n), o resultado dessa última onda de execução pode ser visualizado na figura 2.8.6.

Até que o ponto de destino seja atingido e incluído na “lista fechada”, o algoritmo deverá ser repetido conforme sua descrição. O resultado da execução eventualmente será algo como mostrado na figura 2.8.7.

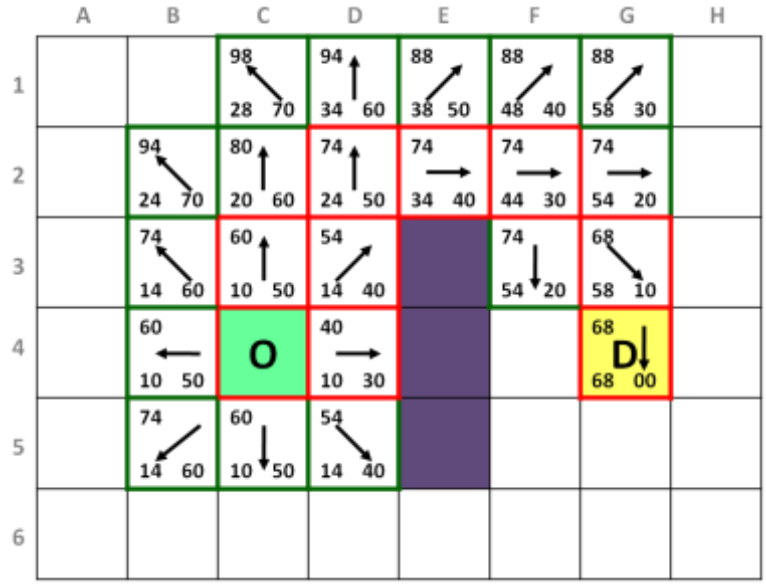


Figura 2.8.7 – Ponto de destino foi atingido e incluído na “lista fechada”.

Como diz o algoritmo, o caminho é encontrado partindo do ponto de destino seguindo os vetores de deslocamento (sentido contrário ao das setas, nas figuras), pulando para o ponto antecessor conforme indicado na “lista fechada”. Esse caminho de pontos é diferente da sequência de exploração e bem mais eficiente. A figura 2.8.8 mostra destacado o caminho encontrado.

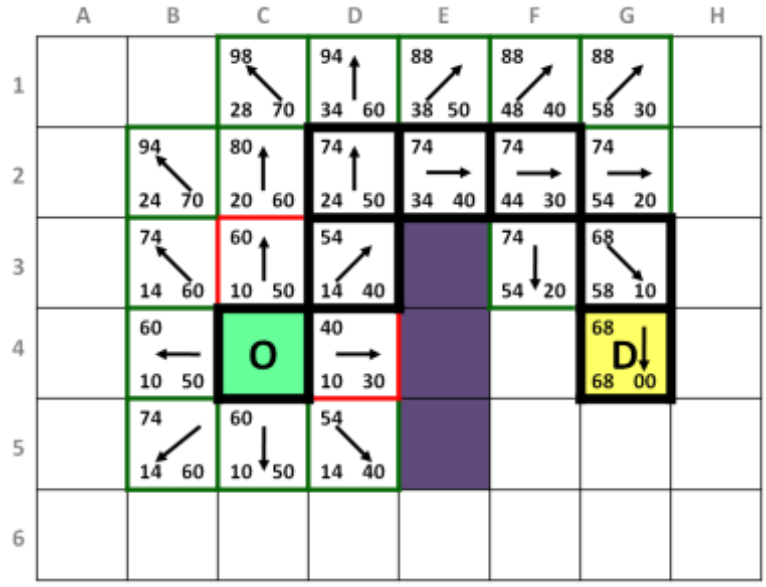


Figura 2.8.8 – Caminho indicado pelos vetores de deslocamento, partindo do destino.

2.9 TELEOPERAÇÃO, TELERROBÓTICA E TELEPRESENÇA

Com vistas a robôs controlados à distância, de acordo com Álvares e Romariz (2002), a telerrobótica é um novo campo de pesquisa que une a robótica com a tecnologia de operação remota cujos meios de transmissão podem ser satélite, rádio, cabo e/ou Internet.

Conforme relatam Rodrigues, Simões e Salazar (2006), a telerrobótica abrange vários setores da sociedade moderna, como linha de montagem de fábricas, a polícia, o exército, os centros de pesquisa espaciais e usinas nucleares, controle de manipuladores robóticos à distância e, também, na medicina ou Telemedicina, mais especificamente na área de Telecirurgia. É possível hoje, com a evolução dessa técnica, acessar locais os mais remotos e realizar cirurgias de alta precisão.

A Telerrobótica é uma forma avançada de teleoperação, é o procedimento pelo qual um operador supervisiona robô(s) através da intermediação de um computador, isto é, o operador localizado fora do ambiente operacional do robô, e com a intermediação de um computador, tem a possibilidade de controle e monitoramento dos dispositivos deste sistema, com o qual este operador pode interagir enviando tarefas a serem cumpridas. Desta forma, o robô controlado executa as tarefas baseado em informações recebidas do operador (humano) e tem a possibilidade de enviar sinais

(dados) para informar ao sistema de controle os parâmetros necessários para obter um controle eficiente.

A *World Wide Web*, ou simplesmente WEB, foi planejada como uma distribuidora de informações, de documentos técnicos, armazenados em grandes sistemas de informações computacionais. Com a evolução da Internet, novas tecnologias foram surgindo e sendo aplicadas nas mais variadas situações. Diante desta evolução, surge a utilização da Internet como meio de controle remoto de robôs.

As vantagens de se utilizar a rede de comunicação Internet como via de controle, devem-se à sua versatilidade em permitir uma série de serviços já existentes e/ou novos que poderão ser criados. Essas características da Internet têm sido favoráveis ao aparecimento de vários sistemas robóticos controlados via Internet. Diversos exemplos existem hoje, tais como: *Telerobot* da Austrália (http://telerobot.mech.uwa.edu.au) que é um sistema robótico teleoperado pela Internet e permite a manipulação de objetos utilizando-se de uma garra. O RobWebCam do GRACO, Grupo de Automação e Controle da Universidade de Brasília, permite o controle remoto de um manipulador para posicionar uma câmera de vídeo transmitindo imagem de vídeo em tempo real (ÁLVARES; ROMARIZ, 2002).

Como observam Álvares e Romariz (1998), a Telerrobótica pode ser definida como sendo uma área da Telemática e da Robótica voltada à teleoperação de sistemas robóticos utilizando-se de um *link* de comunicação. Vale observar que uma nova terminologia está sendo empregada no caso de se utilizar a rede de comunicação Internet como *link* de telecomunicações, neste caso, designando-se de *World Wide Web Robots*, *Internet Robots ou Webot.* No mesmo conjunto de aplicações, há também de se considerar os Avatares Robóticos[2] de Telepresença ou Teleoperação (ADALGEIRSSON, 2009) onde, cada vez mais, os sistemas robóticos precisam ser automatizados de forma a facilitar a interação entre humanos e máquinas.

2.10 CONSIDERAÇÕES FINAIS

Neste capítulo foi mostrado que as ontologias e suas aplicações constituem ferramentas convenientes para organizar e representar as informações e conhecimentos

[2] Avatares Robóticos: genericamente, são dispositivos robóticos móveis utilizados para substituir a presença física humana, capazes de atuar no ambiente como se a pessoa estivesse presente.

necessários para o planejamento e a execução de um sistema em relação aos recursos técnicos oferecidos pelos dispositivos e serviços que se pretende desenvolver. Podem ser úteis também no funcionamento interno do sistema.

É possível organizar e estabelecer uma progressão ou hierarquia entre dados, informação, conhecimento e inteligência. Isto permitirá compreender as formas possíveis e necessárias de processamento de um estágio para atingir o outro e obter os resultados esperados, tanto para o funcionamento interno como externo do sistema.

Os Modelos de Entropia Máxima, também denominados *MaxEnt*, são ferramentas de Aprendizado de Máquina aplicadas ao Processamento da Linguagem Natural. Os Modelos são ajustados ciclicamente por um processo repetitivo para apresentarem estatisticamente resultados o mais próximo possível do que é definido por um banco de dados de texto externo previamente processado geralmente por mãos humanas, denominado *Corpus* de treinamento. Os Modelos são gravados e disponibilizados para utilização posterior.

A teoria dos Autômatos de Estados Finitos é capaz de explicar o funcionamento de sistemas computacionais responsáveis por funcionalidades complexas. A solução resultante pode ser elaborada com a divisão do problema em partes menores.

A Cadeia Oculta de Markov é ferramenta útil e flexível, elaborada a partir de aplicação dos Autômatos de Estados Finitos, capaz de produzir resultados elaborados como a classificação ou identificação de padrões em uma sequência de dados de entrada. Seus parâmetros internos de funcionamento podem ser ajustados indiretamente para obter os resultados desejados. Pode ser associada aos Modelos de Entropia Máxima em variante denominada Modelo de Markov de Entropia Máxima, ou MEMM, que permite a especificação de correlações úteis entre os dados.

Técnica de Inteligência Artificial, os algoritmos de Busca Heurística de Solução são formas computacionalmente possíveis de encontrar solução para dado problema. Poderão ser aplicados para resolver o problema específico de determinar um caminho ou rota de um ponto para outro em uma planta.

Por fim, é possível expandir a utilidade das máquinas inteligentes dotadas da capacidade de comunicação, para sistemas mais abrangentes do ponto de vista da ubiquidade, resultando numa multiplicação de pontos de localização circunscritos por suas áreas de influência. A Teleoperação, a Telerrobótica e a Telepresença são áreas

novas de exploração científica que descortinam um potencial para grandes desenvolvimentos futuros. O raciocínio cibernético pode ser empregado para conceber, analisar, projetar e desenvolver sistemas nesta nova área de conhecimento. O presente trabalho contribui ao abordar problemas, analisar e propor algumas soluções de ordem prática que foram encontrados durante a implementação de um sistema robótico que pode se transformar em sistema telerrobótico sem grandes esforços. Por si, o sistema pode ser considerado um minissistema telerrobótico, uma vez que não há nada que obrigue que seus blocos funcionais estejam localizados fisicamente no mesmo lugar. É possível que as barreiras vencidas para realizar este trabalho não sejam diferentes das que existam para a realização de um sistema de Teleoperação, Telepresença ou Telerrobótica de maior escala.

3. LINGUÍSTICA COMPUTACIONAL

Este capítulo apresenta uma visão geral dos conceitos linguísticos assim como conceitos e abordagens da Linguística Computacional. Com estes eles em mente será possível avaliar melhor o grau de complexidade dos sistemas em função dos recursos que poderá apresentar.

Por se tratar de tecnologias recentes e ainda em desenvolvimento, futuros desenvolvimentos não dispensarão novas pesquisas e revisões bibliográficas. Também, deve-se considerar que o volume de publicações pertinentes a esses assuntos é grande além de rico em termos das abordagens possíveis.

3.1 O PROCESSAMENTO DA LINGUAGEM NATURAL

3.1.1 LINGUAGEM NATURAL

Linguagem Natural ou Língua Natural é aquela usada pelos seres humanos para se comunicarem de forma estruturada. Ela se diferencia de outras também estruturadas como, por exemplo, as linguagens de programação de computador tradicionais, que possuem regras muito restritas para sintaxe e permitem muito pouca ou nenhuma flexibilidade de variação de maneiras de escrever uma mesma coisa.

A Língua Natural possui liberdade de estruturas, pode ser vaga, subjetiva, as palavras podem ser ambíguas, vinculadas ao contexto em que acontece e as orações podem ser fragmentadas (FROMM; DREWS, 1998).

Durante este trabalho será referida como Linguagem Natural, Língua Natural ou mesmo LN.

3.1.2 PROCESSAMENTO DA LINGUAGEM NATURAL

É possível obter uma representação do significado de uma sentença através de sua forma lógica e de maneira independente de contexto. Os relacionamentos semânticos entre as palavras e as frases assim como os significados de cada palavra

podem ser codificados por essa forma lógica. Depois que o relacionamento semântico entre as palavras for identificado, é possível desconsiderar os significados inviáveis (ALLEN, 1995) (FRANCONI, 2001).

O Processamento da Linguagem Natural consiste em executar um conjunto de análises sobre a informação textual que pode ser agrupado em *quatro* grupos (KNIGHT; RICH, 1993). Barros e Robin (1996) propõem a divisão em *cinco* grupos, onde o grupo adicional *Análise do Discurso* visa melhorar as avaliações da coesão e da coerência, ou seja, a maneira como as frases influenciam no significado das subsequentes.

Processamento Léxico: nesta fase, cada palavra é analisada independentemente do restante do texto e recebe uma categorização sintática. Os demais sinais são tratados posteriormente (pontuação). Também chamado de *Processamento Morfológico*.

Processamento Sintático: nesta fase, é aplicada uma transformação na sequência das palavras, para se identificar como se relacionam entre si. As regras dessa transformação são definidas por uma gramática.

Processamento Semântico: nesta fase, as estruturas de palavras são avaliadas para determinar seu significado. É possível, e geralmente se faz, o descarte das estruturas que representam nenhum significado útil.

Processamento do Discurso: nesta fase, avalia-se a maneira como os significados das estruturas influenciam-se entre si e se existe a necessidade de algum ajuste nos significados resultantes, caso exista mais de uma estrutura (BARROS; ROBIN, 1996).

Processamento Pragmático: nesta fase, as estruturas que representam um significado útil são reavaliadas dentro do contexto em que serão aplicadas (RUSSEL; RUSSEL, 2004). O significado da frase decorre do contexto e da intenção do interlocutor (SANTOS; PELIZZONI, 2005). Aqui não há mais vistas para

a estrutura da própria língua, porém sim para o significado (BARROS; ROBIN, 1996).

O Processamento da Linguagem Natural pode ser representado pela figura 3.1.1 (BARROS; ROBIN, 1996).

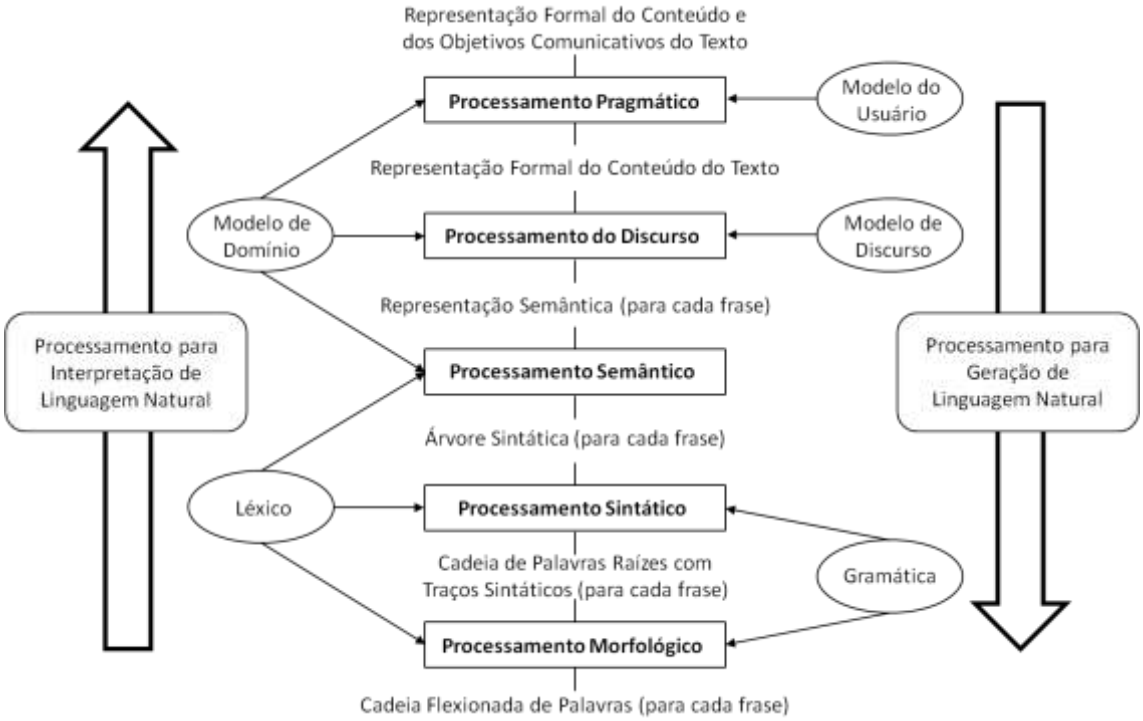

Representação Formal do Conteúdo e dos Objetivos Comunicativos do Texto
Processamento Pragmático
Modelo do Usuário
Representação Formal do Conteúdo do Texto
Modelo de Domínio
Processamento do Discurso
Modelo de Discurso
Representação Semântica (para cada frase)
Processamento para Interpretação de Linguagem Natural
Processamento Semântico
Processamento para Geração de Linguagem Natural
Árvore Sintática (para cada frase)
Léxico
Processamento Sintático
Cadeia de Palavras Raízes com Traços Sintáticos (para cada frase)
Gramática
Processamento Morfológico
Cadeia Flexionada de Palavras (para cada frase)

Figura 3.1.1 – Diagrama de um Processamento de Linguagem Natural genérico.
Fonte: (BARROS; ROBIN, 1996)

Segundo os mesmos autores, as Bases de Conhecimento são arquivos externos que contém informações e regras necessárias ao Processamento da Linguagem Natural. As Bases de Conhecimento constantes da figura 3.1.1 são:

Léxico: conteúdo linguístico e é composto de dicionários sobre os termos onde os termos (ou palavras) são referenciados por suas características morfológicas, sintáticas e semânticas.

Gramática: conteúdo linguístico e é composta de regras que estabelecem quais combinações de palavras são válidas para a língua.

Modelo de Discurso: conteúdo linguístico e é utilizado para contextualização *textual*, ou seja, informações sintáticas e semânticas sobre as frases processadas e que poderão ser usadas para auxiliar na interpretação ou resolução de referências a entidades mencionadas.

Modelo de Usuário: conteúdo sobre o usuário do sistema e á utilizado para contextualização *interpessoal*, ou seja, informações sobre usuário como seus objetivos, planos, intenções, funções, status, conhecimento do domínio etc.

Modelo de Domínio: conteúdo sobre o domínio específico da aplicação e é utilizado para contextualização *enciclopédica*, ou seja, conhecimentos sobre as entidades, relações, eventos, lugares e datas do domínio.

3.2 PROCESSAMENTO LÉXICO OU MORFOLÓGICO

Também aparece na literatura técnica como Análise Léxica ou Análise Morfológica. É a fase que se ocupa da estrutura, forma ou constituição das palavras, através dos morfemas, e as leis que as regem.

Esses morfemas são os elementos linguísticos mínimos que possuem significado e podem ser classificados em lexicais e gramaticais. Os morfemas lexicais possuem significado próprio (por exemplo: sol, sabor e alegria) ao passo que os morfemas gramaticais derivam o significado das relações e categorias da língua (por exemplo: “s” em ruas no final da palavra para marcar plural e “a” em menina para marcar feminino).

Também conhecidos com o nome de radicais, os morfemas lexicais permitem gerar palavras da mesma família que sustentam um significado comum (por exemplo: “branc” em branca e branco). Os morfemas gramaticais são os responsáveis pelas variações entre as palavras e podem caracterizar uma desinência, um afixo ou uma vogal temática (BARROS; ROBIN, 1996).

Resumindo, conforme esclarece Ferreira (2010), os morfemas são classificados em:

Radical ou semantema: é a elemento que detém o significado e a parte comum a todas as palavras de uma mesma família. Há palavras não compostas de unidades menores e são terminadas por consoante ou vogal tônica (por exemplo: sol, mar, jacaré).

- **Livr**o, **livr**aria, **livr**eiro
- **Ferr**o, **ferr**eiro, **ferr**agem

Afixos: são adicionados ao radical para formar palavras novas. Subdividem-se em:

Sufixos: colocados após o radical.

- Mort**al**, leal**dade**

Prefixo: colocados antes do radical.

- **In**feliz, **des**leal

Tema: é o radical acrescido da vogal temática (v.t.). Existem palavras sem tema (atemáticas), que somente possuindo radical, já que são indivisíveis (por exemplo: sol, mar, feliz, javali).

- Canto (radical: cant + v.t.: **o**)
- Livro (radical: liv + v.t.: **o**)

Desinência: terminações indicadoras de flexão, que podem ser dos tipos:

Desinência nominal: indicam flexão do gênero e do número nos nomes (substantivos, adjetivos, numerais e pronomes).

- Garotos: -o = desinência de gênero (masculino); -s = desinência de número (plural).

Desinência verbal: indicam flexão do modo, do tempo, do número e da pessoa nos verbos.

- Estudá_va_mos: -va = desinência de modo temporal (pretérito imperfeito do indicativo); -mos = desinência número pessoal (1ª pessoa do plural).

Vogal temática: colocada após o radical para formar o tema dos nomes e verbos.

Para os verbos, a vogal caracteriza a conjugação a que ele pertence, ou seja, “a”,”e” e “i”.

- Amar = 1ª conjugação; vender = 2ª conjugação; partir = 3ª conjugação.
- O verbo “pôr” é da 2ª conjugação.

Para os nomes, a vogal é átona e acrescentada ao final de substantivos e adjetivos, ou seja, “a”, “e” e “o”.

- Cas**a**, pent**e**, livr**o**s.

Vogal e consoante de ligação: colocados entre o radical e o sufixo para facilitar a pronúncia.

- Cafezal (café + z + al) – **z** é consoante de ligação.
- Cronômetro (cron + o + metro) – **o** é vogal de ligação.

O resultado do Processamento Léxico são as palavras classificadas em suas categorias sintáticas (KNIGHT; RICH, 1993). A expressão “da Maria” pode ser analisada da seguinte maneira:

- “da” – composta pela *preposição* “de” e pelo *artigo* “a”.
- “Maria” – é um *substantivo próprio* “Maria”.

3.2.1 AMBIGUIDADES LÉXICAS

As ambiguidades léxicas das palavras podem ser sintáticas ou semânticas (KROVETZ; CROFT, 1992).

Trata-se de uma *ambiguidade sintática* quando uma palavra puder ser classificada em mais de uma categoria sintática (KROVETZ; CROFT, 1992). Por exemplo, na frase “Quem casa quer casa” a palavra “casa” aparece duas vezes com significados diferentes: um *verbo* e um *substantivo*. Ou seja, são duas categorias sintáticas distintas e, portanto, ambiguidade léxica.

Trata-se de uma *ambiguidade semântica* a condição em que uma palavra apresentar mais de um significado. As ambiguidades semânticas podem ser (FERNANDES, 2010) e (KROVETZ; CROFT, 1992):

Homônimos: palavras com significados distintos, mas apresentam a mesma grafia ou a mesma pronúncia.

Homógrafos: mesma grafia e pronúncia diferente.

- Eu *olho* (verbo) / O *olho* (substantivo).
- Vai *colher* batatas (verbo) / A *colher* (substantivo).

Homófonos: mesma pronúncia e grafia diferente.

- *Acender* (pôr fogo) / *ascender* (subir).
- *Acento* (sinal gráfico) / *assento* (encosto).

Homônimos perfeitos: grafia e pronúncia iguais.

- Eu *caminho* (verbo) / O *caminho* (substantivo).
- A *manga* da camisa (substantivo) / A *manga* madura ("fruta" substantivo).

3.3 PROCESSAMENTO SINTÁTICO

As regras que governam a formação de uma frase e a ordem de seus componentes em uma determinada língua são estudadas pela Sintaxe. A estrutura dessas frases é data pela categoria sintática das palavras que a compõem (BARROS; ROBIN, 1996). Assim, as palavras são analisadas relativamente às demais que compõem a oração (LEVINE; DRANG; EDELSON, 1988). O processamento sintático avalia os resultados da análise morfológica para criar uma estrutura capaz de organizar a frase. Ou seja, ela é transformada de uma lista de palavras para uma unidade organizada com maior coerência gramaticalmente (KNIGHT; RICH, 1993).

Encontrar essa estrutura sintática da frase, ou seja, o resultado de uma análise sintática, é considerada etapa fundamental da interpretação da linguagem natural. É partir dessa estrutura que o processamento semântico poderá ser feito (BARROS; ROBIN, 1996). Com isso, a análise semântica terá menor quantidade de componentes para tratar e a carga computacional será menor. A análise semântica é executada com maior carga computacional que a análise sintática, já que precisará extrair conclusões sobre os componentes da frase (KNIGHT; RICH, 1993).

A ordem dos componentes de uma frase é determinada por regras sintáticas que indicam a localização das palavras em função de sua categoria sintática. Esses componentes podem ser organizados de forma hierárquica entre si (BARROS; ROBIN, 1996). Assim, para estudo da estrutura das frases é adotada uma nomenclatura formal para suas funções e categorias. E por razões práticas, são adotadas também abreviaturas para esses nomes, que são usadas nas análises práticas. Alguns elementos de constituição de frases, com suas abreviaturas, são:

Tabela 3.3.1 – Algumas categorias gramaticais constituintes de frases.
Fonte: (BARROS; ROBIN, 1996) (LEVINE; DRANG; EDELSON, 1988) (WINSTON, 1987)

Nome do Componente	Abreviatura
Oração	O
Frase	F
Sintagma[3] Nominal[4]	SN
Sintagma Verbal[5]	SV
Sintagma Preposicional	SP
Sintagmas Preposicionais	SPS
Substantivo	SUBS ou SUB
Adjetivo	ADJ
Advérbio	ADV
Verbo	V ou VERBO
Determinante	DET
Preposição	PREP

Para se identificar a categoria sintática das palavras durante o processamento sintático, é possível utilizar um glossário. Além dele, as regras sintáticas estabelecidas por uma gramática precisam ser avaliadas para se chegar à compreensão da linguagem natural. Assim, para estabelecer como as palavras de uma frase estão relacionadas entre si, é possível utilizar uma Gramática Livre de Contexto.

3.3.1 GRAMÁTICA LIVRE DE CONTEXTO

Genericamente, um conjunto de regras que definem a formação de palavras e a estruturação de sentenças de uma língua é o que se denomina de gramática (BOUILLON, 1998). Segundo ALLEN (1995), uma gramática deve ser:

a) *genérica* o suficiente para estabelecer o maior número possível de sentenças válidas;
b) *seletiva* o suficiente para identificar os casos problemáticos;
c) *inteligível* o suficiente para favorecer o melhor entendimento de suas regras através da maior simplicidade possível.

Segundo Nunes (1999), uma gramática pode ser representada através de diversos formalismos. Um deles é a gramática de constituintes imediatos ("*Phrase-*

[3] Sintagma é um grupo de palavras que desempenham uma função na frase conforme a categoria sintática da palavra no está no seu núcleo (BARROS; ROBIN, 1996).

[4] Sintagma Nominal tem como palavra no núcleo um ou mais substantivos e desempenham funções de sujeito ou objeto (BARROS; ROBIN, 1996) (GONZALES; LIMA, 2001).

[5] Sintagma Verbal possui no núcleo um verbo ou uma locução verbal (BARROS; ROBIN, 1996).

Structure Grammar" – PSG), definida por um conjunto de quatro subconjuntos representado pela "4-upla" ou quádrupla <T,N,P,S>, onde:

T = conjunto de palavras da língua (também denominados elementos terminais);

N = conjunto de categorias funcionais e de categorias lexicais (elementos não terminais);

P = conjunto de regras de produção; e

S = o símbolo inicial pertencente a N.

Comentam Vieira e Lima (2001) que os modelos de gramática situados entre os livres de contexto e os sensíveis ao contexto, propostos por diversos pesquisadores, são geralmente os mais indicados, inexistindo, porém, um formalismo considerado como o melhor.

As gramáticas livres de contexto são conjuntos de regras gerais de estruturação de uma língua independentemente do contexto em que acontecem (GRISHMAN, 1986). São construídas em torno da ideia simples de se executar substituições subsequentes de componentes estruturais de uma frase, tanto para componentes intermediários da análise quando para componentes terminais, sendo que o elemento a ser substituído pelos elementos da direita é colocado à esquerda. (BARROS; ROBIN, 1996) (WINSTON, 1987).

Por exemplo, para um trecho qualquer em Português, é possível generalizar regras de uma gramática livre de contexto (LEVINE; DRANG; EDELSON, 1988):

Tabela 3.3.2 – Exemplo de gramática livre de contexto.

Fonte: (LEVINE; DRANG; EDELSON, 1988)

1	F	→	SN, SV
2	SN	→	DET, SUB
3	SN	→	SUB
4	SV	→	V, SN
5	SUB	→	menino | banana
6	DET	→	o
7	V	→	comeu

A regra 1 significa que a frase F se compõe de um sintagma nominal SN e um sintagma verbal SV:

$$F \rightarrow SN, SV.$$

As regras 2 e 3 significam que um sintagma nominal se compõe de duas formas, ou seja, de um determinante DET e um substantivo SUB ou então somente de um substantivo:

$$SN \rightarrow DET, SUB$$

$$SN \rightarrow SUB$$

A regra 4 significa que um sintagma verbal SV se compõe de um verbo V e um sintagma nominal SN:

$$SV \rightarrow V, SN$$

Os elementos gerais que podem compor uma frase em Português estão presentes nas quatro primeiras regras, que utilizam elementos não terminais. As outras regras, ou seja, cinco, seis e sete, envolvem elementos terminais, que são as palavras propriamente ditas. Então é possível fazer a análise sintática, por exemplo, da frase "O menino comeu banana", usando a gramática livre de contexto apresentada pela tabela 3.3.1. Isso pode ser feito da forma conforme a seguir.

Inicialmente se aplica a regra 1, que diz que a frase é composta de um sintagma nominal e um sintagma verbal. Em seguida se aplica a regra 2, uma vez que ela representa melhor a composição do elemento não terminal SN. Não se aplica a regra 3, pois não se ajusta a nenhum caso neste momento. Depois, ao se aplicar a regra 4, será obtido que o elemento não terminal SV é composto de um verbo e um sintagma nominal. Na sequencia faz sentido aplicar a regra 3, que diz que o elemento não terminal SN se compõe de um substantivo. O processo continua adiante da mesma forma, até a substituição completa dos elementos não terminais pelos equivalentes terminais segundo a gramática proposta. Para conhecer se a frase está sintaticamente conforme a gramática, a tabela 3.3.3 resume a aplicação dessas regras, passo a passo.

Tabela 3.3.3 – Exemplo da aplicação de uma gramática livre de contexto.

1º passo	F	
2º passo	SN	SV
3º passo	DET SUB	SV
4º passo	DET SUB	V SN
5º passo	DET SUB	V SUB
6º passo	o SUB	V SUB
7º passo	o menino	VSUB
8º passo	o menino	comeu SUB
9º passo	o menino	comeu banana

Outra maneira comum de representar a estrutura sintática pode ser vista na figura 3.3.1. Ela também representa a mesma gramática livre de contexto. Cada ramificação ou galho mostra a substituição de um elemento não terminal pelos seus respectivos componentes gramaticalmente corretos. O processo é repetido recursivamente, da mesma forma como descrito pouco antes, até o ponto em que todos os componentes não terminais sejam substituídos por componentes terminais. Portanto, se o último nível da árvore são os elementos terminais desta gramática, o primeiro nível é seu elemento inicial.

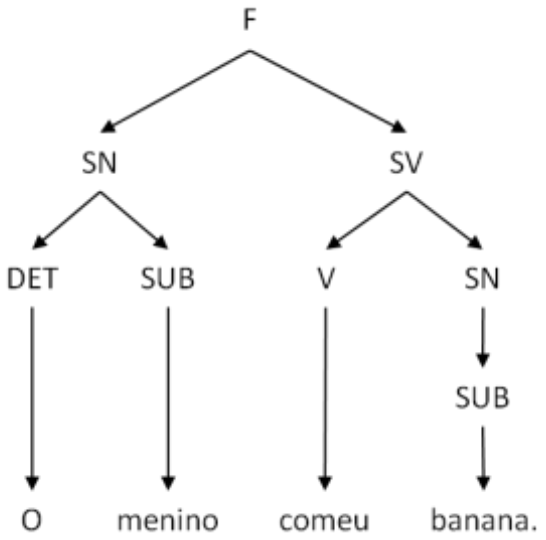


Figura 3.3.1 – Análise sintática de uma frase com estrutura em árvore.
Fonte: (LEVINE; DRANG; EDELSON, 1988)

3.3.2 ANÁLISE TOP-DOWN *E ANÁLISE* BOTTOM-UP

Para executar a análise sintática de uma frase e obter sua estrutura, há duas estratégias que derivam seu nome de uma observação da representação em árvore: análise *Top-Down* e análise *Bottom-Up*. O resultado dessas análises é o mesmo, ou seja, todas as palavras da frase são associadas a sua função conforme a gramática.

Análise *Top-Down*: a sequência de aplicação das regras da gramática se inicia no elemento inicial descendo até que os todos os elemento terminais sejam efetivamente as palavras da frase em análise (KNIGHT; RICH, 1993).

Por exemplo, se for utilizada a gramática da tabela 3.3.4 para a frase “Maria foi ao cinema.” será obtida uma análise representada em árvore conforme mostra a figura 3.3.2.

Tabela 3.3.4 – Exemplo de regras de gramática.

Fonte: (LEVINE; DRANG; EDELSON, 1988)

1	F	→	SN, SV
2	SN	→	DET, SUB
3	SN	→	SUB
4	SV	→	V, SN

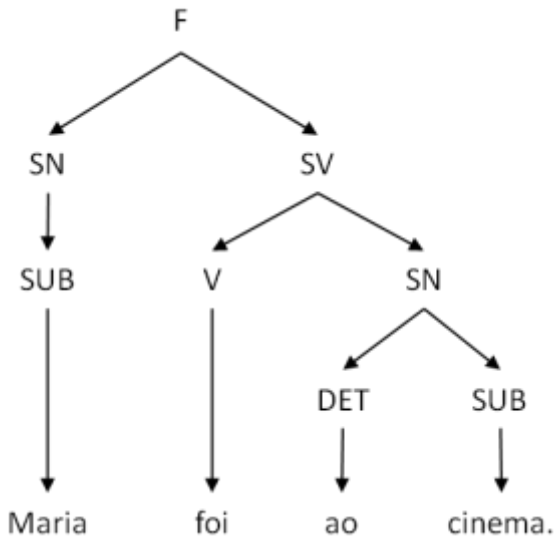


Figura 3.3.2 - Análise *Top-Down* do exemplo e estrutura representada em árvore.

Se as regras da gramática a tabela 3.3.4 forem transcritas como mostra a tabela 3.3.5, é possível aplicar a mesma gramática em outro exemplo e fazer a análise *Top-Down* da frase “O gato caçou o rato”. O resultado será a sequência passo a passo mostrada na tabela 3.3.6 (PEREIRA, 2006).

Tabela 3.3.5 – Transcrição da gramática anterior.

1	Frase	→	Sujeito Predicado
2	Sujeito	→	Artigo Substantivo
3	Predicado	→	Verbo Objeto
4	Predicado	→	Verbo Artigo Substantivo

Tabela 3.3.6 – Passo a passo da análise *Top-Down* da frase "O gato caçou o rato".

1º passo	Frase	
2º passo	Sujeito	Predicado
3º passo	Artigo Substantivo	Predicado
4º passo	o Substantivo	Predicado
5º passo	o gato	Predicado
6º passo	o gato	Verbo Artigo Substantivo
7º passo	o gato	caçou Artigo Substantivo
8º passo	o gato	caçou o Substantivo
9º passo	o gato	caçou o rato

Análise *Bottom-Up*: a sequência de aplicação das regras da gramática acontece ao contrário, ou seja, ela se inicia nos elementos terminais em direção ao elemento inicial. Esta análise está completa quando os elementos terminais são as palavras da frase e o nó superior é o elemento raiz da gramática.

Usando como exemplo tanto a última frase quando a gramática anteriores e fazendo uma análise *Bottom-Up*, o resultado será como mostra a tabela 3.3.7.

Tabela 3.3.7 – Análise *Bottom-Up* da frase "O gato caçou o rato".

1º passo	o gato caçou o rato
2º passo	Artigo gato caçou o rato
3º passo	Artigo Substantivo caçou o rato
4º passo	Sujeito caçou o rato
5º passo	Sujeito Verbo o rato
6º passo	Sujeito Verbo Artigo rato
7º passo	Sujeito Verbo Artigo Substantivo
8º passo	Sujeito Predicado
9º passo	Frase

3.4 PROCESSAMENTO SEMÂNTICO

O processamento semântico tem por objetivo extrair o significado da frase que teve sua estrutura revelada pelo processamento sintático (LEVINE; DRANG; EDELSON, 1988). Aborda o significado sem considerar quem originou ou como a frase é usada (VIEIRA; LIMA, 2001).

O principal resultado do processamento semântico é ter o significado da frase representado ou transformado de alguma forma útil pelo computador. Outro resultado importante é estabelecer restrições sobre essa representação de significado, em

decorrência das relações existentes entre a estrutura sintática e a estrutura semântica da frase (KNIGHT; RICH, 1993).

A análise semântica trata de questões mais difíceis que a sintaxe. Seus métodos e resultados são menos exatos, sendo uma área de estudo que aborda problemas cujos contextos são incertos e complexos de serem qualificados. Isso decorre do fato de que o significado de uma palavra está ligado ao conhecimento de mundo e a questões mais obscuras como estados mentais e consciência (VIEIRA; LIMA, 2001).

Outra questão que a análise semântica necessita resolver são as ambiguidades formal-ortográficas. Sendo de dois tipos, elas podem ser ambiguidades lexicais ou ambiguidades sintáticas (VIEIRA; LIMA, 2001). Quando uma palavra possui mais de um significado, isso é conhecido como ambiguidade lexical. Considerando os exemplos abaixo:

- Maria está sentada no **banco** da praça.
- João pagou a conta de telefone no **banco**.

É possível observar que a palavra banco foi usada com significados diferentes em cada uma das frases. Na primeira frase, trata-se de um objeto de se sentar e na segunda, uma instituição financeira. Um objeto de se sentar pode possuir variadas formas e tamanhos, com a maior parte dos seus atributos concretos. Já uma instituição financeira possui a maior parte dos seus atributos abstratos.

É possível resolver os problemas de ambiguidade lexical analisando a frase como um todo, já que as demais palavras ao redor geralmente colaboram para estabelecer o seu significado próprio.

Quando a interpretação de uma frase apresenta mais de um significado possível, isso constitui uma ambiguidade sintática. Considerando o exemplo a seguir:

- Todas as meninas gostam de uma boneca.

É possível observar que a frase pode ser interpretada de duas maneiras diferentes. Uma em que cada menina gosta de uma boneca diferente e outra em que as meninas gostam de uma mesma boneca.

Para que um sistema seja capaz de determinar o significado de uma frase, um conjunto de regras e procedimentos de processamento precisa ser estabelecido. São essas regras e procedimentos que deverão ser capazes de retornar o que uma frase

significa. Usando como exemplo o analisador semântico proposto por Levine, Drang e Edelson (1988) a frase abaixo pode ser processada com as regras a seguir.

- O menino comeu banana.

Regra 1: se a primeira parte de uma frase for composta de um determinante seguido de um substantivo, este substantivo é denominado sujeito.

Regra 2: se logo depois do sujeito existir um verbo, ele expressará a ação executada pelo sujeito.

Regra 3: se depois do sujeito e do verbo, na sequência for encontrado um substantivo, este último é denominado objeto.

Regra 4: se uma frase apresentar os elementos estruturais na ordem sujeito, verbo e objeto, é possível saber qual ação o sujeito tomou sobre o objeto.

Com essas regras, um sistema computacional será capaz de extrair a informação de qual foi a ação do menino em relação à banana. Esta informação pode ser então utilizada em algum processamento posterior ou armazenada na forma de conhecimento, se for o caso. O conteúdo semântico de uma frase pode ser extraído ou representado por meio de diferentes métodos. A seguir será visto o que é denominado de Gramática Semântica.

3.4.1 GRAMÁTICA SEMÂNTICA

Uma gramática semântica é uma gramática livre de contexto, onde os elementos não terminais correspondem aos conceitos envolvidos no domínio de uma dada aplicação ao invés das categorias sintáticas (WONG; MOONEY, 2005). Uma gramática semântica é similar à gramática sintática. Também utiliza a substituição sequencial de elementos não terminais e que pode ser representada por uma estrutura em árvore. A principal diferença é que na gramática semântica as ramificações não correspondem à função sintática das palavras. Referem-se à forma da linguagem e são usadas para (de)codificar as regras semânticas específicas do uso que se faz daquela língua (GRANBERG; GEORGSSON, 2010). Cada regra da gramática semântica, ou seja, cada substituição corresponde a uma parte do significado (GE; MOONEY, 2006).

Um exemplo de aplicação de gramática semântica específica para um contexto pode ser visto em (WONG; MOONEY, 2005). Trata-se do ATIS (*Air Travel*

Information Service, em tradução livre seria algo como Serviço de Informação sobre Viagem Aérea), onde um usuário faz consultas sobre voos utilizando linguagem natural. Considerando a consulta abaixo:

- "*Show me the flights that stop in Boston*"
  (Mostre-me os voos que param em Boston)

É possível representar sua estrutura em árvore como na figura 3.4.1.

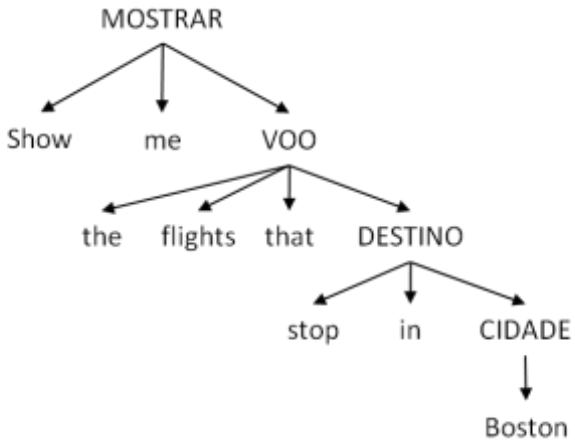


Figura 3.4.1 – Um exemplo de estrutura em árvore aplicada ao ATIS.
Fonte: (MILLER *et al.*, 1994).

No exemplo, a primeira regra da gramática é a que associa o elemento terminal *Boston* ao elemento não terminal CIDADE que neste caso é um significado útil no contexto da utilização: [CIDADE → *Boston*].

Outra regra é a que associa o trecho "*stop in x"* (param em x) ao elemento não terminal DESTINO, estabelecendo o significado, ou conceito, de destinação: [DESTINO → "*stop in*" CIDADE].

Vale observar que essas regras são aplicadas de baixo para cima, ou seja, *bottom-up*, iniciando pelas substituições simples como a constante [CIDADE → *Boston*] para depois aplicar as regras mais elaboradas como [DESTINO → "*stop in*" CIDADE] até o topo da estrutura em árvore.

Com o uso desta gramática semântica, a frase pode ser traduzida para os conceitos usados no domínio da aplicação a que se destina. No exemplo, o sistema deverá fazer uma busca por voos com destino até a cidade informada.

3.4.2 DEPENDÊNCIA CONCEITUAL

Para representar o conhecimento sobre eventos expressos por frases, de forma independente da linguagem natural na qual foi expresso para permitir a realização de inferências, Roger Schank (SCHANK, 1975) propôs a teoria da Dependência Conceitual (DC). Para isto, utiliza um conjunto de primitivas conceituais que podem ser combinadas para compor os significados de qualquer linguagem natural. Primitivas usadas para criar as representações das ações são mostradas na tabela 3.4.1.

Tabela 3.4.1 – Ações Primitivas – ACT.

Fonte: (BARR; FEIGENBAUM, 1986)

AÇÃO - *ACT*	**SIGNIFICADO**
	Atos caracterizados por efeitos físicos
GRASP	Domínio físico de um objeto por um ator (p. ex., agarrar)
EXPEL	Expulsão algo do corpo de um animal ou ator (p. ex., chorar)
INGEST	Ingestão de algo por um animal ou ator (p. ex., comer)
MOVE	Movimentação de parte do corpo (p. ex., chutar)
PROPEL	Aplicação de força a um objeto físico (p. ex., empurrar)
	Atos caracterizados pela mudança de estado resultante
ATRANS	Mudança em relação abstrata, relativo a um objeto (p. ex., dar, possuir)
PTRANS	Mudança de local de um objeto (p. ex., ir, correr, colocar)
	Atos usados como instrumento para outros atos
ATTEND	Direcionamento de um órgão de sentido para um estímulo (p.ex. escutar)
SPEAK	Produção de sons (p. ex., falar)
	Atos caracterizados por efeitos mentais
MBUILD	Construção de novas informações a partir de antigas (p. ex., decidir)
MTRANS	Transferência de uma informação mental (p. ex., contar, ver, ouvir)

Ao lado das primitivas ACT, Schank propõe outros elementos para representar as conceitualizações:

- PP (ou *Picture Producer*, em inglês): atores, objetos físicos, forças naturais ou divisões da memória humana (processador do consciente, memória intermediária e de longo termo);
- AA (ou *Action Aiders*, em inglês): modificadores ou atributos de ações; e
- PA (ou *Picture Aiders*, em inglês): modificadores ou atributos de PP.

São conceitualizações básicas:

a) Um ator (PP) executando uma primitiva ACT; e

b) Um objeto (PP) com a descrição de seu estado (PA).

Não são as palavras da linguagem em si que são representados. Os elementos primitivos de conceitualização refletem o pensamento subjacente à linguagem (BARR; FEIGENBAUM, 1986). As relações semânticas entre esses conceitos subjacentes são representados nas dependências entre conceitualizações de forma independente da linguagem. A figura 3.4.2 mostra essas representações conforme proposição de Schank.

As informações da linguagem natural relativas aos tempos, modos e aspectos verbais podem ser incluídas nas representações de eventos (ações) por um conjunto de tempos conceituais proposto por Schank mostrado na tabela 3.4.2 (KNIGHT; RICH, 1993).

Tabela 3.4.2 – Tempos Conceituais.

Fonte: (KNIGHT; RICH, 1993)

Modificador	**Significado**
p	Passado
f	Futuro
t	Transição
t_s	Iniciar transição
t_f	Transição encerrada
k	Continuidade
?	Interrogação
/	Negação
nil (nenhum)	Presente
Delta	Infinito
c	Condicional

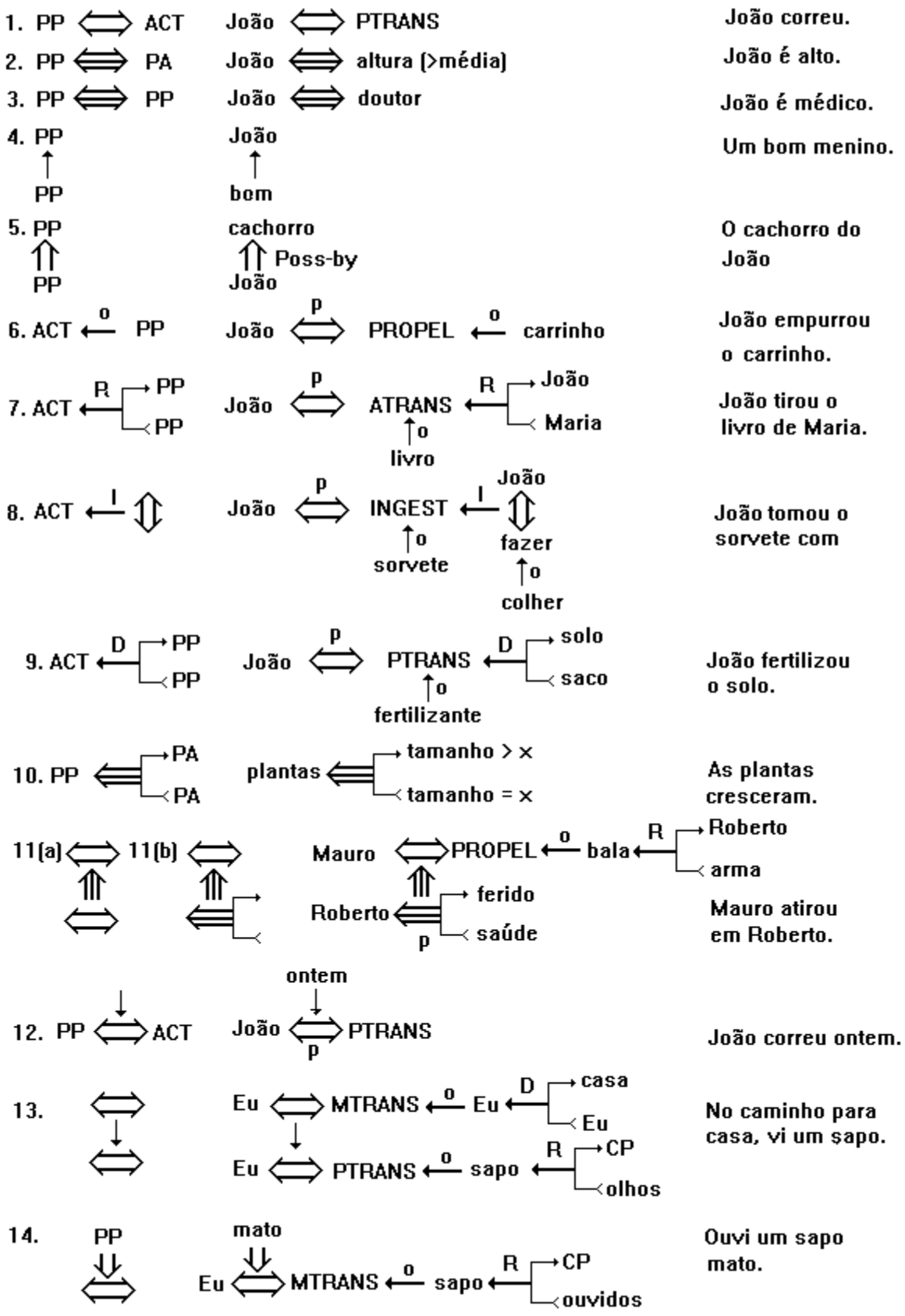


Figura 3.4.2 – Dependência entre Conceitualizações.

Fonte: (KNIGHT; RICK, 1993)

A interpretação das representações da figura 3.4.2 é mostrada na tabela 3.4.3.

Tabela 3.4.3 – Interpretação das Dependências entre Conceitualizações.

Fonte: (KNIGHT; RICH, 1993)

Regra	Relação entre
1	Ator (PP) e a ação (ACT) que ele causa. Como nem o ator nem a ação podem ser considerados primários, a dependência é bidirecional.
2	Objeto (PP) e um atributo (PA) que o descreve. Em DC, muitas descrições de estado são representadas como escalas numéricas.
3	Dois objetos (PP), um dos quais pertence ao conjunto definido pelo outro.
4	Objeto (PP) e um atributo (PP) que já tinha sido seu predicado. A seta aponta para o PP que está sendo descrito.
5	Dois objetos (PP), um dos quais fornece uma informação sobre o outro, sendo que as mais comuns são: posse (*Poss-by*), localização (*Loc*) e contenção física (*Cont*). A seta aponta para o conceito sendo definido.
6	Ação (ACT) o seu (o)bjeto (PP). A seta aponta para a ação, já que o contexto determina o significado da relação com o objeto.
7	Ação (ACT) e a sua fonte e o (R)ecipiente.
8	Ação (ACT) e o (I)nstrumento com que é executada. O instrumento precisa ser uma conceitualização completa (i.é conter uma ACT) e não ser apenas um simples objeto físico.
9	Ação (ACT) e a sua fonte e o (D)estino físicos.
10	Objeto (PP) e o seu estado inicial e final (mudança de estado).
11	Uma conceitualização e outra que a motiva. As formas apresentadas descrevem a causa de uma ação e a causa de uma mudança de estado. A seta aponta para a causa.
12	Evento e o momento em que ocorreu.
13	Uma conceitualização e outra que é o momento da primeira. O exemplo ilustra a forma de tratar processamento das informações humanas: ver é representado como uma transferência de informação entre os olhos e o processador do consciente (CP). CP é uma abstração utilizada em DC.
14	Uma conceitualização e o lugar em que ocorre

A frase “Já que fumar pode matar quem fuma, Leda parou de fumar.” pode ser representada, em Dependência Conceitual, conforme a figura 3.4.3 (KNIGHT; RICK, 1993).

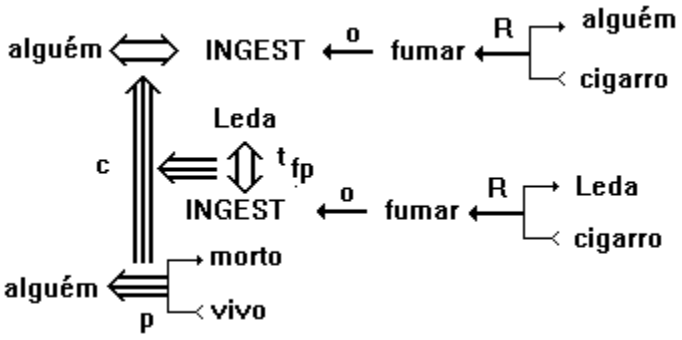


Figura 3.4.3 – Exemplo de representação usando Dependência Conceitual.
Fonte: (KNIGHT; RICK, 1993)

O elo de causalidade vertical significa "fumar pode matar alguém". O modificador *c* torna a relação de causa em uma possibilidade.

A ligação de causalidade horizontal indica que esta primeira causa (elo vertical) é o motivo que fez Leda parar de fumar. A qualificação t_{fp} especifica que a dependência existente entre Leda e INGEST parou e que isto (a parada) aconteceu no passado.

As vantagens do uso da Dependência Conceitual são:

a) Dispõe de um conjunto de primitivas que representam o significado independente da linguagem natural;
b) Representa adequadamente eventos normalmente presentes em frases em linguagem natural;
c) Utiliza uma única regra de inferência para cada primitiva de ação representada (e não uma para cada palavra); e
d) A representação da ação contém, por si só, várias inferências, pois é estruturada pela dependência entre conceitualizações.

Contudo, a Dependência Conceitual é uma teoria para a representação de ações relativamente simples. A representação devidamente formalizada de expressões como "João apostou cinquenta mil com Manuel que o São Paulo seria campeão brasileiro" ocuparia o espaço de duas páginas (SCHANK; OWENS *Apud* KNIGHT; RICH, 1993). Isto se deve à utilização de primitivas de muito baixo nível, exigindo grande esforço de detalhamento na formalização.

Outra desvantagem é que ela é apenas uma teoria da representação de eventos. Um modelo genérico de representação do conhecimento precisa ser capaz de representar mais coisas além de eventos. Com mecanismos de inferência mais especializados, representações em DC podem ser vistos como instâncias de redes semânticas, como sugerem Lehnert (1978 *Apud* KNIGHT; RICH, 1993) e Schank e Carbonell (1979 *Apud* KNIGHT; RICH, 1993).

A teoria da Dependência Conceitual visa tornar intuitivo o processamento da linguagem humana e não prover suporte a programas que processem linguagem natural. Ainda que as frases originais possuam ambiguidades, uma representação produzida por esta teoria é única e sem ambiguidades, com o mesmo conteúdo conceitual.

3.4.3 ESQUEMA DE CASOS

Para representar eventos, Schank (1975) propôs a dependência conceitual. Com ações primitivas básicas, todas as ações podem ser reduzidas a uma sequência de estruturas baseadas na ideia de gramática de caso. Os constituintes do evento são representados por casos ou papéis relativos às primitivas do evento (GANGEL, 1985).

A ideia central proposta por Schank de se utilizar um esquema de casos, ações primitivas e papéis relevantes compõe o núcleo de uma análise de orações de ação fornecidas como entrada de um sistema. Para os objetivos do sistema proposto, é suficiente entender que um Esquema de Caso é uma lista de atributos (casos ou papéis) dependentes do verbo de ação para o qual vários tipos de informação (valores ou parâmetros) podem estar acompanhando (GANGEL, 1985).

Segundo Gangel (1985) um Esquema de Caso supõe que um conjunto de casos pode ser usado para definir a estrutura profunda (significado, e não a estrutura superficial da oração) da linguagem natural. O sistema representativo de Schank baseia-se no relacionamento conceitual entre objetos e ações. Num Esquema de Caso, o significado não depende da forma da oração. As seguintes orações possuem a mesma estrutura profunda:

- "Alberto deu o livro para Beatriz"
- "Beatriz recebeu o livro de Alberto"
- "O livro foi dado por Alberto para Beatriz"

Generalizando as conceitualizações e primitivas propostas por Schank, é possível reduzir as orações acima em um único esquema de caso (GANGEL, 1985)(SPECIA; RINO, 2002):

EVENTO
ATOR: Alberto
AÇÃO: Transferência (de propriedade)
OBJETO: o livro
DIREÇÃO: DE: Alberto
PARA: Beatriz

Onde: ATOR, AÇÃO, OBJETO e DIREÇÃO estão vinculados aos seus respectivos constituintes "Alberto", "Transferência", "o livro" e “DE: Alberto PARA: Beatriz”.

Os constituintes vinculados à ação (ou estado) na estrutura conceitual da DC desempenham papéis definidos que correspondem aos casos do Esquema de Casos. Os papéis conceituais da DC são (SPECIA; RINO, 2002):

1) ATOR: quem realiza uma ação;
2) AÇÃO: ação causada a um objeto por um ator;
3) OBJETO: item passivo sobre a qual a ação é realizada;
4) RECIPIENTE: receptor do resultado de uma ação;
5) DIREÇÃO: localização na qual uma ação é direcionada;
6) ESTADO: estado em que um objeto está; e
7) INSTRUMENTO: instrumento por meio do qual uma ação ocorre.

Chama-se Gramática de Casos (FILLMORE, 1968a e 1968b) a teoria de representação semântica baseada nas relações existentes entre a ação (ou estado) associada ao verbo e seus demais constituintes. Essas relações são chamadas de relações de casos ou casos conceituais. Mesmo tendo sido desenvolvida em base puramente linguística, foi utilizada em vários sistemas de PLN e foi fundamento para novas teorias linguísticas, como a própria Dependência Conceitual (DC) (SPECIA; RINO, 2002).

Fillmore supõe que as línguas humanas são restritas e que as relações entre os constituintes de uma oração se enquadram em pequeno número de tipos. Eles caracterizam os chamados casos conceituais que podem ser identificados pelo julgamento que seres humanos fazem acerca dos acontecimentos ao seu redor. Fillmore

propõe seis casos conceituais ou papéis semânticos (SPECIA; RINO, 2002)(WALTER, 1990):

1) AGENTIVO (A): instigador da ação (indicada pelo verbo), normalmente ser animado. Exemplo: João em "João abriu a porta".

2) INSTRUMENTAL (I): força ou objeto inanimado envolvidos na ação ou estado (indicados pelo verbo). Exemplo: A chave é o Instrumental em "João abriu a porta com a chave" e “A chave abriu a porta”.

3) DATIVO (D): ser animado afetado pela ação ou estado (indicados pelo verbo). Exemplo: João é o Dativo em "Nós convencemos João de que ele venceria".

4) FACTIVO (F): objeto resultante da ação ou estado (indicados pelo verbo ou compreendido como parte do significado do verbo). Exemplo: O Factivo em "Nós fizemos para João uma Jaqueta" é a Jaqueta. O Sonho é o Factivo em "João teve um sonho sobre Maria".

5) LOCATIVO (L): localização ou orientação espacial da ação ou estado (indicados pelo verbo). Exemplo: Manaus é o Locativo em "É úmido em Manaus".

6) OBJETIVO (O): qualquer coisa representada por um substantivo cujo papel na ação ou estado (indicados pelo verbo) é determinado pela interpretação semântica do próprio verbo. Exemplo: A Porta é o Objetivo em "A chave abriu a porta" e "João abriu a porta com a chave".

De acordo com Fillmore, cada sentido expresso por um verbo associa-se a um esquema de caso que descreve o conjunto de papéis obrigatórios, opcionais, sua ordem ou o que mais for sintaticamente preciso, quando aquele significado é usado na oração. Por exemplo, o esquema de caso para o significado normal da palavra "abrir" diz que um papel Objetivo é obrigatório, mas os papéis Agentivo ou Instrumental são opcionais (WALTER, 1990)(GANGEL, 1985).

Assim, uma oração cujo verbo principal é "abrir" requer um constituinte para o papel Objetivo (algo que é "aberto") e pode ou não possuir constituintes para o papel do Agentivo (alguém que realizar a "abertura") e para o papel do Instrumental (um objeto com o qual se realiza a "abertura"). Causa estranhamento evidente a frase "João

abriu com a chave", uma vez que falta o constituinte para o papel do Objetivo, obrigatório para a palavra "abrir". (WALTER, 1990).

Colocando entre parêntesis os papéis opcionais e a ordem dos papéis é a de preferência, os esquemas de casos para os verbos “Dar” e “Matar” podem ser representados (GANGEL, 1985):

- Dar - [(Agente) (Beneficiado) Objeto]
- Matar - [(Agente) Objeto (Instrumento)]

Os verbos significam uma mudança de estado ou atividade perpetrada por um agente. Ambos possuem regras sintáticas são diferentes. Dar, na oração "João deu à Maria um livro" possui um Beneficiado (Maria). Mas em Matar, não: "João matou a Maria um livro", onde existe claramente um erro. Matar precisa de um Objeto e não um Beneficiado. Assim "João matou a Maria" constitui a estruturação sintática correta (GANGEL, 1985).

3.5 PROCESSAMENTO DO DISCURSO E PROCESSAMENTO PRAGMÁTICO

A questão principal tanto da Pragmática quanto da Análise do Discurso é encontrar o significado de uma frase ou parágrafos (BARROS; ROBIN, 1996).

O *Processamento Pragmático* refere-se ao uso da língua do ponto de vista da interação social, ou seja, o efeito que a comunicação tem sobre os participantes da comunicação. Portanto, não mais se preocupa com a formação estrutural da língua, como fazem a sintaxe e a semântica (BARROS; ROBIN, 1996). Aborda o significado considerando quem originou e como a frase é usada (VIEIRA; LIMA, 2001).

O *Processamento do Discurso* procura identificar a influência de uma frase sobre as demais, sendo importante na interpretação de pronomes (por exemplo: eu, você, ela, este, aquela etc.) e de dêiticos (por exemplo: hoje, aqui, agora etc.) (BARROS; ROBIN, 1996).

Neste caso, uma dificuldade típica é a resolução de uma referência anafórica, ou seja, uma citação a um elemento anterior através do uso de pronomes (KNIGHT; RICH, 1993). Por exemplo, considerando as frases abaixo:

- Maria comprou uma boneca.
- Ela está entusiasmada.

É possível observar a existência de uma referência anafórica na segunda frase. O pronome pessoal “ela” faz referência a “Maria” citada na primeira frase. Portanto, o significado da segunda frase pode ser que “Maria está entusiasmada com a boneca que comprou” se for considerado o conhecimento indicado pela referência anafórica.

Outro caso típico é a resolução de uma referência catafórica, ou seja, citar de um elemento posterior que fornecerá significado a ser considerado. Considerando, por exemplo, a frase:

- Este é o nosso lema: *servir bem para servir sempre*.

É possível observar que o pronome demonstrativo “Este” se refere à frase “servir bem para servir sempre”, que até então não foi mencionando da sentença.

A pragmática procura determinar se o significado de uma estrutura sintática é de fato o mais apropriado dentro do contexto em que é utilizado (PEREIRA, 2006). Uma grande contribuição nesta análise vem da Filosofia da Linguagem e são os Atos da Fala (BARROS; ROBIN, 1996).

Atos da Fala, segundo Searle (1971) são as atividades desenvolvidas pelos falantes de uma língua enquanto fazem uso dela. Segundo Barros e Robin (1996) os tipos de atos da fala são relacionados abaixo, seguidos de suas possíveis realizações linguísticas:

- Assertivos → assertiva, constatação, afirmação, negação;
- Diretivos → ordem, comando, requisição;
- Comissivos → promessa, garantia, juramento, aposta;
- Expressivos → elogio, agradecimento, parabenização;
- Declarativos → (a) encontro, indicação, declaração (b) declaração de guerra (c) estímulo, resignação, excomunhão.

Dependendo do contexto em que é utilizado, um mesmo enunciado pode significar diferentes Atos da Fala. Assim, o processamento automático dos Atos da Fala é tarefa difícil, tendo-se em vista que estabelecer a intenção e o contexto dos participantes da comunicação é tarefa obscura (BARROS; ROBIN, 1996).

3.6 CONSIDERAÇÕES FINAIS

Uma sentença tem sua estrutura sintática revelada por processamentos morfológico e sintático. As regras gramaticais, constantes numa gramática, definem a representação dessa estrutura. Para este processamento, também são necessárias outras informações que constam num léxico (ou dicionário), como por exemplo, as categorias morfológicas.

A forma lógica de uma sentença pode ser revelada pelo processamento semântico dessa estrutura sintática, onde o léxico é também necessário por dispor de recursos sobre o significado dos itens lexicais.

Assim, a transformação das representações de significado de uma sentença para se obter a sua forma lógica depende dos recursos disponíveis na gramática e no léxico.

O desenvolvimento de interfaces capazes de processar a língua natural requer conhecimentos diversos e específicos da língua. O Processamento de Linguagens Naturais, como o Português, envolve, por exemplo, formação das palavras, regras gramaticais da formação das frases e orações e a extração do significado de uma frase analisando suas palavras e estrutura. O contexto em que as frases são usadas também deve ser considerado durante a extração do significado.

Com o uso dos conhecimentos adequados é possível desenvolver tais interfaces. Para o reconhecimento da Língua Portuguesa será preciso dotar o sistema de conhecimentos específicos, que podem ou não ser aproveitados de outras línguas.

As teorias Esquema de Casos ("*Case Frames*") proposta por Fillmore e Dependência Conceitual ("*Conceptual Dependency*") proposta por Schank apresentam recursos que podem ser estendidas através de generalização e especialização (inclusão de recursos de detalhamento) para se adaptarem ao contexto de utilização e, assim, constituir ferramentas de processamento mais convenientes ao emprego que se deseja.

4. O SISTEMA ROBÓTICO PROPOSTO

O propósito deste capítulo é apresentar o sistema robótico. À medida que as partes integrantes que o compõem forem sendo descritas, serão tecidos comentários e avaliações sobre o contexto de decisão sobre soluções, tecnologias e técnicas utilizadas ou consideradas para aplicação.

Este sistema possui como objetivo primordial permitir que um usuário humano informe instruções relativas a tarefas ou a ações que deseja que um robô execute, usando simplesmente linguagem natural em Português. O usuário descreverá de forma natural, sem necessidade de utilizar termos técnicos ou expressões previamente aprendidas por treinamento específico, o que deseja que seja executado, através de uma Interface em Linguagem Natural. Após finalizar a descrição do que deseja ser realizado, o sistema processará as orações e frases visando extrair os comandos, os parâmetros e a sequência suficientes para cumprir com a tarefa instruída, conforme os recursos que dispõe.

4.1 VISÃO GERAL E ARQUITETURA

Organizando a estrutura do sistema robótico por blocos funcionais, ele pode ser descrito por quatro componentes, organizados de forma sequencial intuitiva:

- Interface em Linguagem Natural (IFLN);
- Inteligência e Controle;
- Interface com Robô (IFR); e
- Robô.

Um diagrama esquematizando a relação entre os blocos funcionais é mostrado na figura 4.1.1.

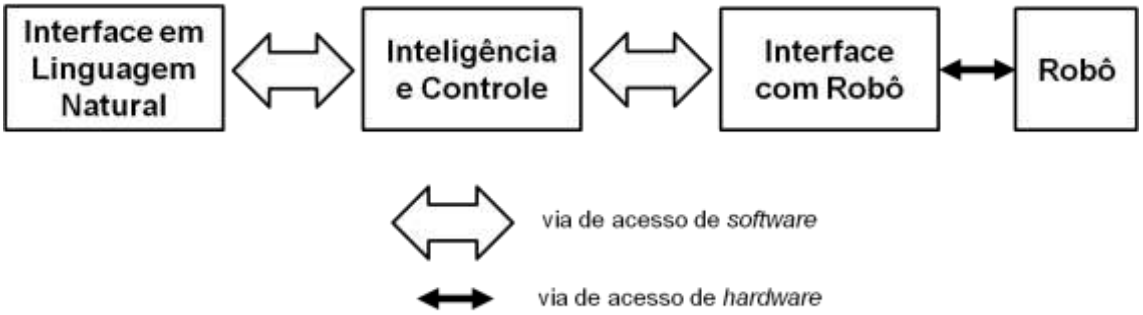


Figura 4.1.1 – Sistema Robótico esquematizado por blocos funcionais.

A Interface em Linguagem Natural é a principal via de interação do usuário com o sistema robótico. A versão desenvolvida para o sistema proposto consiste em uma janela tipo formulário de preenchimento e seus elementos, baseada nos recursos do sistema operacional utilizado (no caso, o recurso do denominado *Windows Forms* disponibilizado pela linguagem C# e a biblioteca de funções disponibilizadas pelo "*dotNet Framework*" da Microsoft). Porém, uma versão para *Web* pode ser também desenvolvida para o sistema, utilizando os recursos disponíveis no "*dotNet Framework*".

O bloco de Inteligência e Controle é um conjunto de funções que desempenham o processamento crítico fundamental do Sistema Robótico proposto. É neste bloco funcional que ocorrem o processamento da linguagem natural, o sequenciamento dos comandos identificados e o controle do robô propriamente dito. Independentemente da interface que seja utilizada para comunicação com o usuário, estas funções sempre serão utilizadas da mesma forma para realizar o processamento e o trabalho.

A Interface com o Robô é uma via de *software* que o bloco de Inteligência e Controle dispõe para comunicação com o robô e executar os comandos. As interfaces elétricas existentes para acionamento estão acionáveis somente para ela. O *software* é o único intermediário possível para o bloco de Inteligência e Controle. Os detalhes elétricos, canais de comunicação e seus protocolos são abstraídos neste ponto através de bibliotecas de função preferencialmente desenvolvidas e fornecidas pelo fabricante do Robô, como é o caso do Curumim.

O Robô é o ponto focal das capacidades disponibilizadas pelo sistema. É através dos recursos e funções de ação, no ambiente físico, presentes no robô que os comandos e serviços reconhecidos pela linguagem natural podem ser planejados. O

sistema proposto dispõe de um robô em *hardwar*e (o Curumim). Uma indicação de posição e orientação do robô na planta do ambiente oferece a funcionalidade de virtualização, desenvolvida e integrada na Interface em Linguagem Natural. Porém, outras variações desta interface podem não dispor desta funcionalidade, como seria o caso de uma para *e-mail* ou telefone celular, por exemplo. A possibilidade de virtualização agregou certa conveniência ao sistema ao permitir estudo e simulação prévia de experimentos.

4.1.1 VISÃO DETALHADA DO SISTEMA PROPOSTO

Tendo visto de maneira geral organização do sistema proposto, é possível detalhar os outros níveis de agrupamento das funções que desempenham seu funcionamento e controle. A figura 4.1.2 representa com mais detalhes o conjunto de sub-blocos do Sistema Robótico e como estão organizados em relação aos demais.

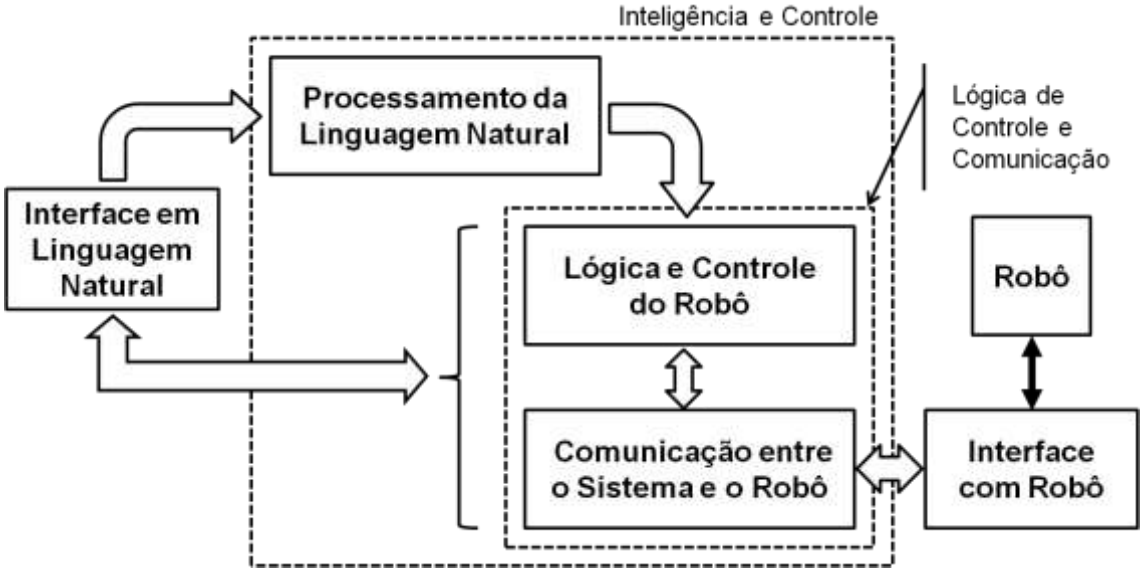

Inteligência e Controle
Processamento da Linguagem Natural
Lógica de Controle e Comunicação
Interface em Linguagem Natural
Lógica e Controle do Robô
Robô
Comunicação entre o Sistema e o Robô
Interface com Robô

Figura 4.1.2 – Sistema Robótico esquematizado em detalhes.

É possível observar que a Interface em Linguagem Natural possui duas vias de comunicação com o sistema, ambas com o bloco de Inteligência e Controle. A primeira é com o bloco de Processamento da Linguagem Natural, para as instruções vindas do usuário. A segunda é o bloco de Lógica e Controle de Comunicação, para permitir as

demais funcionalidades do sistema bem como complementar o funcionamento da própria interface.

O bloco de Processamento da Linguagem Natural comunica-se somente com o bloco de Lógica de Controle e Comunicação uma vez que o resultado produzido pelo primeiro é a sequência de comandos e parâmetros interpretados a partir do texto do usuário.

Internamente, o bloco de Lógica e Controle de Comunicação é composto por dois outros blocos de funções, *Lógica e Controle do Robô* e *Controle da Comunicação entre o Sistema e o Robô*, detalhados pelas seções 4.3.2 e 4.3.3 a seguir.

A Interface com o Robô comunica-se com o bloco de *Lógica de Controle e Comunicação* por via de *software* para permitir o controle e supervisão do Robô, com o qual se comunica por via de *hardware*.

O bloco de Processamento da Linguagem Natural também poderia ser denominado de Inteligência Artificial, ou Inteligência de Máquina, já que é implementado usando técnicas desta área de conhecimento. Também agrupa as funções que oferecem recursos inteligentes ao sistema, como o algoritmo de busca heurística de caminho, associado ao comando Rotear, abordado nas seções 5.2.1 (abstrações agregam utilidade) e 2.8 (algoritmo de busca do melhor caminho). É possível mencionar também a detecção do comando Rotacionar implícito em instruções de deslocamento lateral, conforme o caso do Robô utilizado. O próprio processamento estocástico da linguagem natural contido neste bloco utiliza técnica de aprendizado de máquina.

Ele poderá receber, em futuros desenvolvimentos, outras funções para melhorar o Processamento da Linguagem Natural, novo serviços agregados ao Sistema Robótico ou outras funcionalidades que venham a ser úteis.

4.1.2 ESQUEMATIZAÇÃO GENERALIZADA DO SISTEMA

Uma vez que a concepção inicial do sistema robótico proposto seria permitir mais de uma Interface em Linguagem Natural e mais de uma Interface com Robô, uma esquematização generalizada dessa concepção é mostrada na figura 4.1.3.

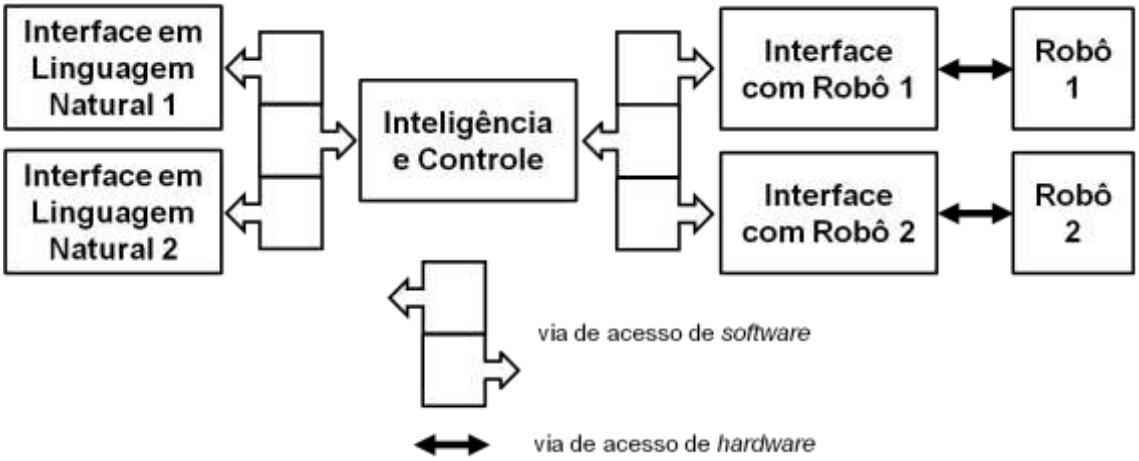


Figura 4.1.3 - Sistema Robótico com esquematização generalizada.

Na versão generalizada haveria uma via ou barramento comum de acesso entre as Interfaces de Linguagem Natural (IFLN) e o bloco de Inteligência e Controle como também há uma via ou barramento comum de acesso entre este e as Interfaces com Robô (IFR). Uma limitação do sistema como proposto atualmente é que as mesmas tarefas solicitadas serão executadas indistintamente por todos os robôs e as interfaces deverão ser usadas uma por vez. Uma sugestão de desenvolvimento futuro seria a concepção da parte do *software* de Inteligência e Controle para gerenciar solicitações diferentes de cada usuário, via sua IFLN, para serem executadas por cada robô, via sua própria IFR.

4.2 A INTERFACE EM LINGUAGEM NATURAL

Uma das contribuições deste trabalho e a principal via de interação do usuário com o sistema robótico será apresentada a seguir. Com ela, será possível fazer referência e trazer esclarecimentos sobre as partes internas do sistema robótico, seu funcionamento e, assim, abordar também a interface robótica.

A interface em linguagem natural consiste em uma janela tipo formulário de preenchimento e seus elementos, baseada nos recursos do sistema operacional utilizado (no caso, o recurso do denominado *Windows Forms* disponibilizado pela linguagem C# e a biblioteca de funções disponibilizadas pelo "*dotNet Framework*" da Microsoft).

Para ilustrar melhor, a figura 4.2.1 apresenta uma visão geral de sua aparência e a disposição de seus elementos.

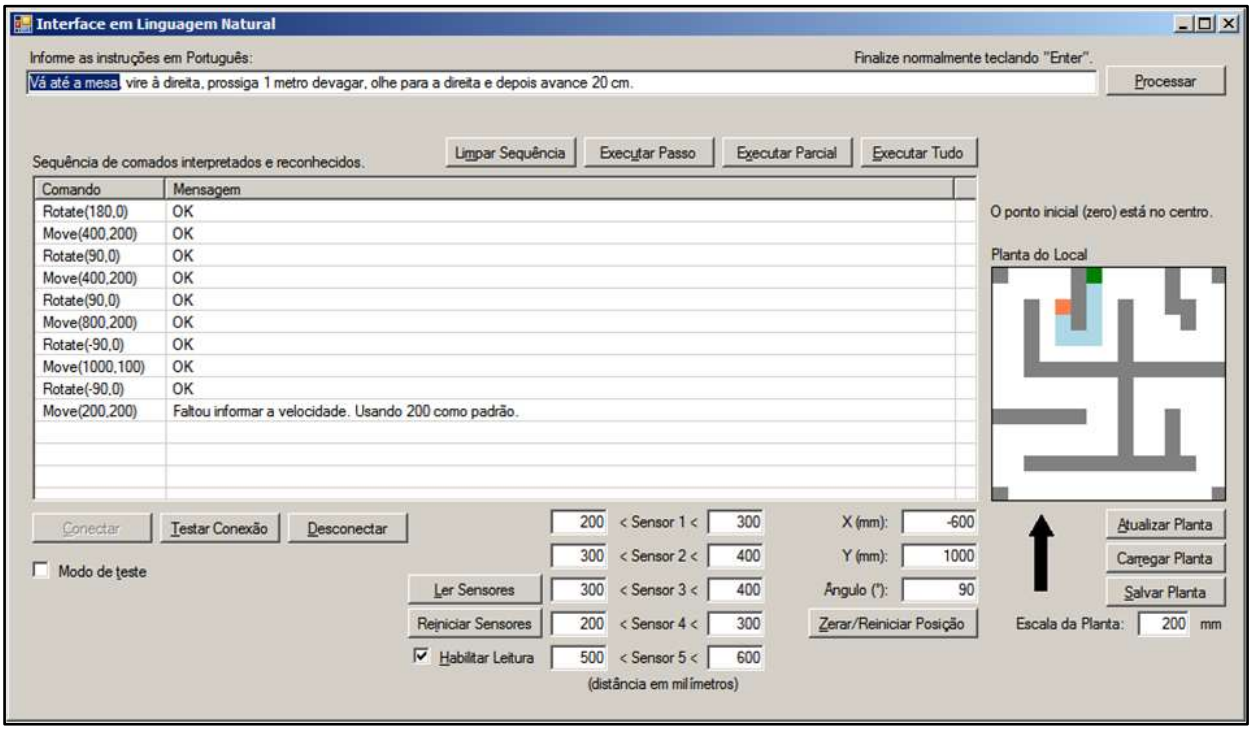


Figura 4.2.1 – Visão geral da Interface em Linguagem Natural.

Sua finalidade é a de permitir que o usuário informe as instruções que deseja, utilizando linguagem natural em Português, apresentar os comandos que foram interpretados à partir dessas instruções, controlar a execução destes comandos e permitir ajuste em algumas funcionalidades do sistema robótico.

Nesta interface, oura contribuição deste trabalho, e um recurso visual de comunicação, é a Planta ou Mapa do local onde o robô se movimenta. Esta planta permite ao usuário uma maneira mais conveniente de interação para controle ou acompanhamento do funcionamento do sistema, pois representa diretamente várias informações internas: onde o sistema “acredita” ser a localização do robô, os obstáculos reconhecidos pelo sistema, é de onde o sistema irá buscar as soluções de caminho ou suas alternativas e a forma pela qual o usuário poderá avaliar ou acompanhar esta solução ou alternativa. A letra “E” do item 4.2.1 a seguir aborda mais sobre este recurso.

4.2.1 ELEMENTOS QUE CONSTITUEM A INTERFACE

A interface em linguagem natural foi composta com alguns elementos para permitir o controle da utilização dos recursos de linguagem natural e também o próprio sistema robótico. A seguir será mostrado o conjunto desses elementos.

A) Instrução em Português (campo de texto para linguagem natural)

É o campo de texto onde o usuário poderá inserir as orações contendo as instruções em Português, e em linguagem natural, que serão interpretadas pelo sistema. Na figura 4.2.2, este campo é indicado pela letra "A", ilustrando onde está localizado e como se apresenta.

B) Sequência de comandos interpretados e reconhecidos

É uma caixa de listagem aonde são exibidos os comandos interpretados e reconhecidos a partir das instruções em Português. Os comandos são organizados na ordem de execução como se encontram no texto do usuário. Além dos comandos propriamente ditos, na coluna, ao lado direito, são incluídas mensagens de erro ou informações complementares associadas ao comando da mesma linha, localizado na coluna da esquerda.

Na figura 4.2.2, este campo é indicado pela letra "B", ilustrando onde está localizada esta caixa de listagem e como se apresenta.

C) Botões de controle da execução dos comandos

Durante a utilização, e para controle da maneira de execução da sequencia de comandos, foram incluídos quatro botões, cada com finalidades próprias. São eles:

- Limpar Sequência: remove todos os comandos da lista;
- Executar Passo: executa somente o comando atual. Se for pressionado outra vez, executará o comando seguinte e assim por diante. Serve para execução passo a passo da sequência;
- Executar Parcial: executa os passos até o final da sequencia a partir do comando escolhido; e
- Executar Tudo: executa toda a sequência de comandos, a partir do primeiro comanto até o último.

Uma última forma de solicitar a execução de um comando qualquer da lista é clicando duas vezes sobre ele na lista.

A localização e aparência destes botões na Interface em Linguagem Natural é indicada na figura 4.2.2 com o grupo de elementos da letra “C”.

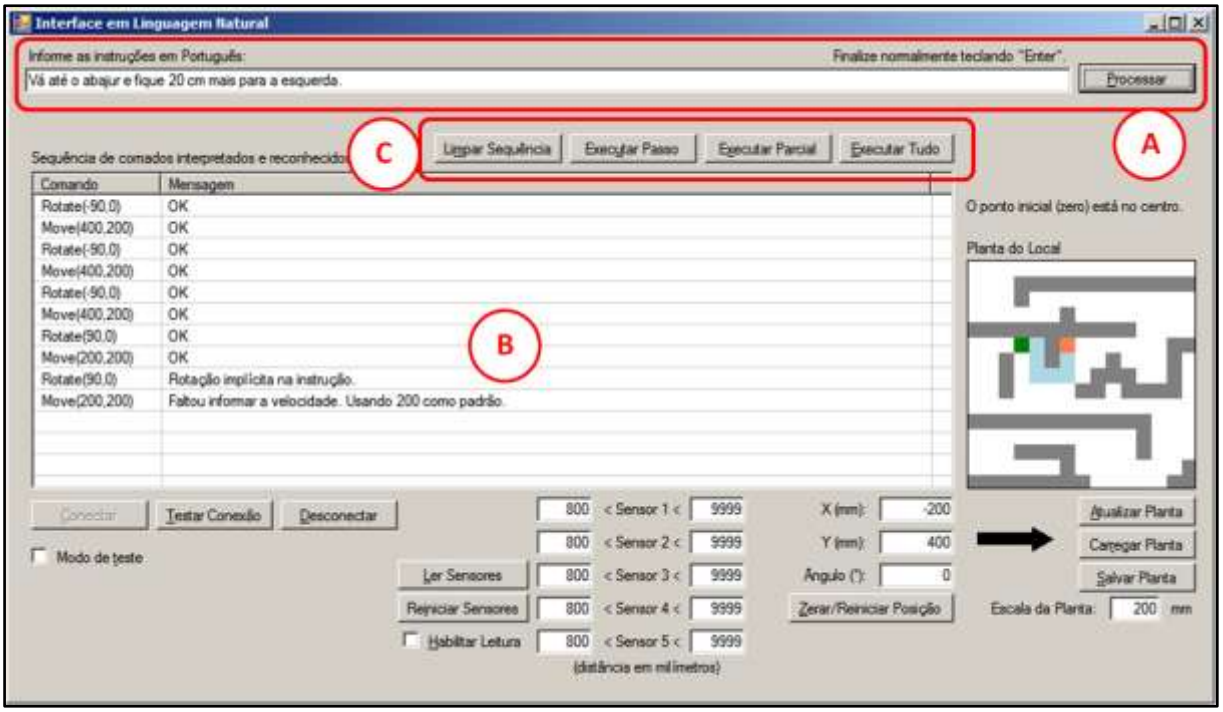


Figura 4.2.2 – Principais elementos da Interface em Linguagem Natural.

Foram apresentados acima os elementos fundamentais para permitir a utilização dos recursos de linguagem natural do sistema robótico proposto. A seguir serão mostrados elementos úteis no controle de outras funcionalidades do sistema robótico.

D) Indicadores da localização e da orientação do robô

Há três caixas de texto que exibem o conteúdo dos registradores internos de localização e orientação do robô. Essas coisas de texto são interativas, ou seja, à conveniência do usuário, é possível alterar o conteúdo desses registros internos, para corrigir algum erro no experimento, ou estabelecer alguma condição inicial desejada para um teste, por exemplo.

Para a localização do robô, são usadas coordenadas cartesianas X e Y. Para sua orientação, é usado o ângulo de rotação em relação ao referencial inicial.

O referencial inicial, ou zero, está localizado no centro do mapa. As convenções utilizadas são a cartesiana e a trigonométrica. Ou seja:

- Deslocamentos positivos em X são para a direita;
- Deslocamentos negativos em X são para a esquerda;
- Deslocamentos positivos em Y são para cima;
- Deslocamentos negativos em Y são para baixo;
- Ângulos de rotação positivos são no sentido anti-horário; e
- Ângulos negativos são no sentido horário.

Apesar de o sistema prever ângulos com precisão de um grau, por razões de simplificação, o funcionamento interno do sistema trabalha somente com ângulos retos, ou seja, so006Dente em passos de 90 graus (positivos ou negativos).

Também, foi adicionado um botão para atribuir facilmente todos os registradores à situação de zero (localização em X, em Y e ângulo de rotação), logo abaixo das caixas de texto.

- Zerar/Reiniciar Posição: este botão reestabelece os registradores internos de localização e de orientação do robô para o ponto inicial zero, ou seja, coordenadas X, Y e ângulo de rotação no valor zero.

Na figura 4.2.3, a localização das caixas de texto com a representação do conteúdo dos registradores internos de localização e rotação, bem como o botão "Zerar/Reiniciar Posição" está indicada pelo número "3".

E) Mapa ou planta de representação do local

Para permitir uma representação do local aonde se localiza o robô e onde são realizados os experimentos, foi colocada uma planta. Ela possui função interativa com o sistema robótico, de onde ele poderá retirar informações úteis para realizar a navegação pelo local, por exemplo.

Esta planta pode ser personalizada através de cliques com o cursor do *mouse*, introduzindo ou removendo obstáculos reais, por exemplo. As plantas criadas podem ser guardadas em arquivos no computador hospedeiro e também recuperadas posteriormente à conveniência do usuário.

A localização do robô na planta é indicada por um quadrado alaranjado.

Quando uma instrução é fornecida, que mencione um destino ou alvo de referência para navegação, este ponto é indicado por um quadrado verde. O percurso proposto para navegação é indicado em azul-claro e os obstáculos, em cinza-escuro.

O ajuste da escala da planta é feito através de uma caixa de texto denominada "Escala da Planta", inserido com para esta finalidade, sendo que as medidas deverão ser inseridas em milímetros, por conversão.

- Escala da Planta: campo de texto contendo o valor de ajuste da escala da planta. Deve ser preenchido com valores numéricos e em milímetros.

Antes de indicar numa figura a localização e aparência da planta, é útil conhecer outros elementos presentes e relacionados com o a localização e orientação do robô bem como o recurso de armazenagem e recuperação das plantas.

F) Botões de controle da Planta/Mapa

Para permitir que as novas plantas editadas sejam armazenadas no computador hospedeiro, ou então recuperar as plantas previamente armazenadas, foram inseridos botões apropriados.

Também, caso haja alguma falha na atualização da planta em relação aos registradores internos ou alguma alteração manual nos elementos disponíveis, foi incluído um terceiro botão para forçar uma atualização manual da planta conforme as condições de representação interna do sistema.

- Atualizar Planta: botão que redesenha a planta com base nos valores internos, caso necessário;
- Carregar Planta: botão que permite localizar um arquivo no computador hospedeiro contendo uma planta previamente guardada para posterior reutilização; e
- Salvar Planta: botão que permite guardar a planta atualmente em uso em um arquivo no computador hospedeiro para posterior utilização.

G) Referência visual da orientação do robô

Para auxiliar o usuário a identificar a orientação do robô em relação à planta, foi incluída uma referência visual do ângulo de rotação com relação à origem. Trata-se de uma seta ou de uma figura do próprio robô. A opção de uma ou de outra representação pode ser escolhida via *software*.

A localização da Planta do Local, dos botões de controle de suas funcionalidades, do campo de texto para ajuste de sua escala e do símbolo de representação da orientação do robô estão indicados na figura 4.2.3 pelo grupo de elementos de número “4”.

H) Indicação da leitura dos sensores de obstáculo

Um conjunto de caixas de texto foi previsto, para indicar os limites inferior e superior de leitura dos sensores do robô. Por enquanto, o sistema prevê apenas sensores de obstáculo e também um limite de tolerância (erro) na leitura dos mesmos. Assim, caso um obstáculo se apresente dentro do limite previsto, o sistema indicará um erro caso seja instruído um movimento além do limite inferior.

Estas caixas de texto podem ser editadas manualmente para inserir os valores que o usuário desejar. Isto pode ser útil para durante a simulação de alguma condição experimental.

Juntamente com as caixas de texto associadas aos limites de tolerância dos sensores, foram previstos elementos de controle do funcionamento dos recursos de leitura dos referidos sensores e esses elementos serão discutidos no tópico a seguir. Este grupo de elementos é indicado pelo número “2” da figura 4.2.3.

I) Elementos de controle da leitura dos sensores de obstáculo

Em algumas condições de experimentação e teste, pode ser conveniente operar o sistema robótico desativando os recursos de restrições de movimento por obstáculos. Não só a leitura dos sensores é operação lenta de ser executada, como também a geração de mensagens de erro pode se tornar distração inconveniente.

Assim, foi inserida uma caixa de marcar que permite habilitar e desabilitar os recursos e rotinas internas do sistema que estão relacionados com a leitura dos sensores e a restrição de movimento.

- Habilitar Leitura: caixa de marcar que habilita, quando marcada, os recursos de leitura dos sensores de obstáculo automaticamente conforme a natureza do comando. Se desmarcada, desabilitará tais recursos.

Caso seja útil utilizar o sistema sem os inconvenientes da restrição de movimento e das mensagens de erro, basta reiniciar os registradores de leitura dos sensores, preenchendo-os com valores equivalentes aos de um ambiente sem

obstáculos. Para isto, basta inserir os valores manualmente ou então clicar no botão "Reiniciar Sensores" que será comentado logo a seguir. Por outro lado, é possível inserir manualmente valores de restrição e o sistema passará a considerá-los automaticamente.

Caso seja útil ao usuário operar a leitura dos sensores de obstáculo do robô de forma avulsa e manual, foram incluídos dois botões para esta finalidade. São eles:

- Ler Sensores: botão que dispara uma leitura dos sensores do robô; e
- Reiniciar Sensores: botão que retorna os registradores internos de leitura dos sensores para a condição equivalente a um ambiente sem obstáculos.

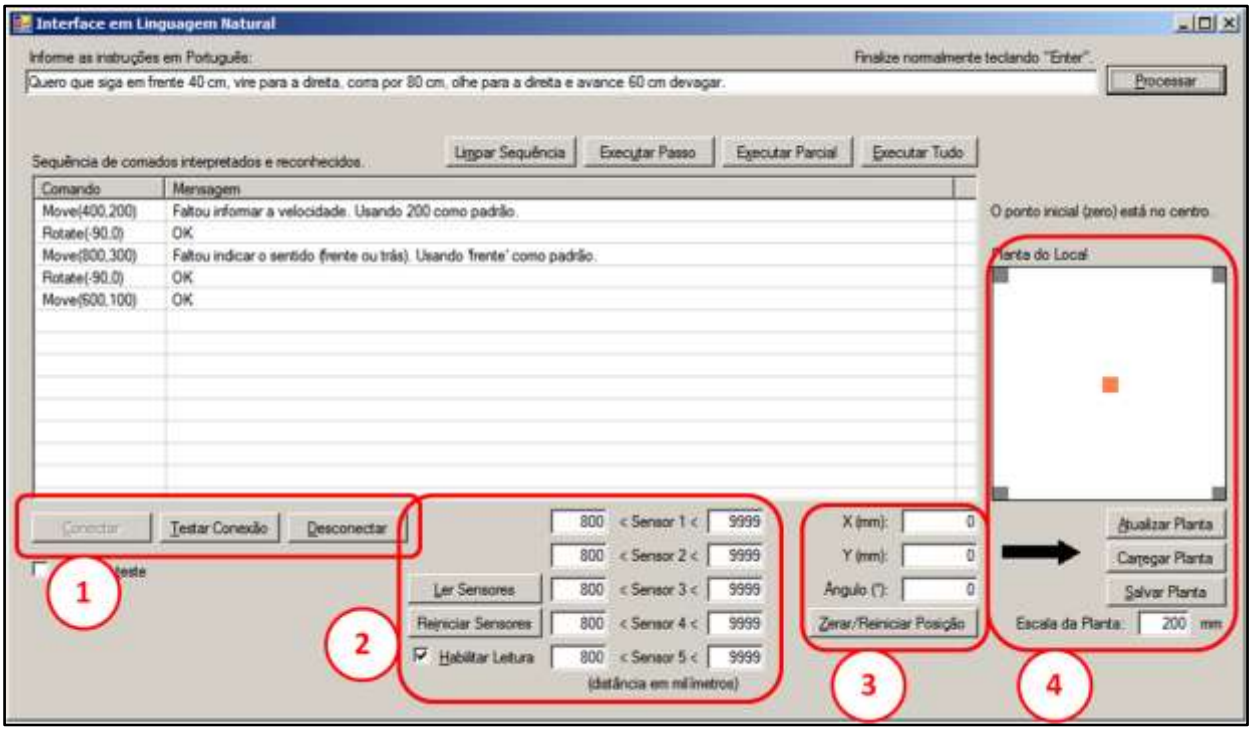


Figura 4.2.3 – Elementos de controle do sistema robótico.

Tanto a caixa de marcar "Habilitar Leitura", quanto os botões de controle da leitura dos sensores, estão indicados no grupo de elementos "2" da figura 4.2.3.

J) Botões de controle da conexão com o robô

Para controlar a conexão do sistema robótico com o robô, e assim permitir que seja possível trabalhar sem que ele esteja presente, foram incluídos três botões para esta finalidade. São eles:

- Conectar: botão que realiza a conexão através do sistema de comunicação;
- Testar conexão: botão que realiza um teste de comunicação com o robô; e
- Desconectar: botão que desfaz a conexão através do sistema de comunicação.

Esses elementos são indicados na figura 4.2.3 pelo grupo número "1".

K) Modo de testes e seus controles de funcionamento

O Sistema Robótico Proposto foi dotado de funcionalidade para registro automático de alguns parâmetros internos de funcionamento em arquivo no disco local do computador hospedeiro. Isto permite que sejam realizadas análises posteriores dos resultados de experimentos de forma mais criteriosa. O Apêndice A.2.2 indica quais são esses parâmetros.

O controle desta funcionalidade é composto pelos seguintes elementos:

a) Campo de texto "Nome do usuário";
b) Caixa de marcar "Modo de teste";
c) Botão "Continuar teste"; e
d) Rótulo para mensagens e avisos durante o teste.

Até que a caixa de marcar "Modo de teste" esteja efetivamente marcada, todos os demais elementos permanecerão invisíveis, para que o visual da interface seja menos congestionado. Uma vez marcada, os demais elementos tornar-se-ão visíveis imediatamente.

O Modo de Teste altera ligeiramente a maneira da Interface em Linguagem Natural funcionar para que seja possível coletar o veredito do usuário sobre o resultado do processamento do sistema. Ou seja, para cada instrução submetida e processada pelo sistema será preciso que o usuário marque os comandos resultantes de interpretação errônea e finalize clicando no botão "Continuar teste".

A caixa de texto "Nome do usuário" é apresentada com um nome padrão pré-preenchido (Visitante). Seu conteúdo pode ser atualizado conforme necessário e determinará o nome do arquivo que será usado para guardar os registros automáticos durante os testes de utilização do sistema.

Durante a utilização no modo de teste, mensagens e avisos específicos serão exibidos no recurso visual denominado rótulo, ou seja, um elemento que se presta bem à exibição de textos curtos apenas para leitura do usuário.

O grupo de elementos de controle do modo de teste é indicado pelo número "5" na figura 4.2.4.

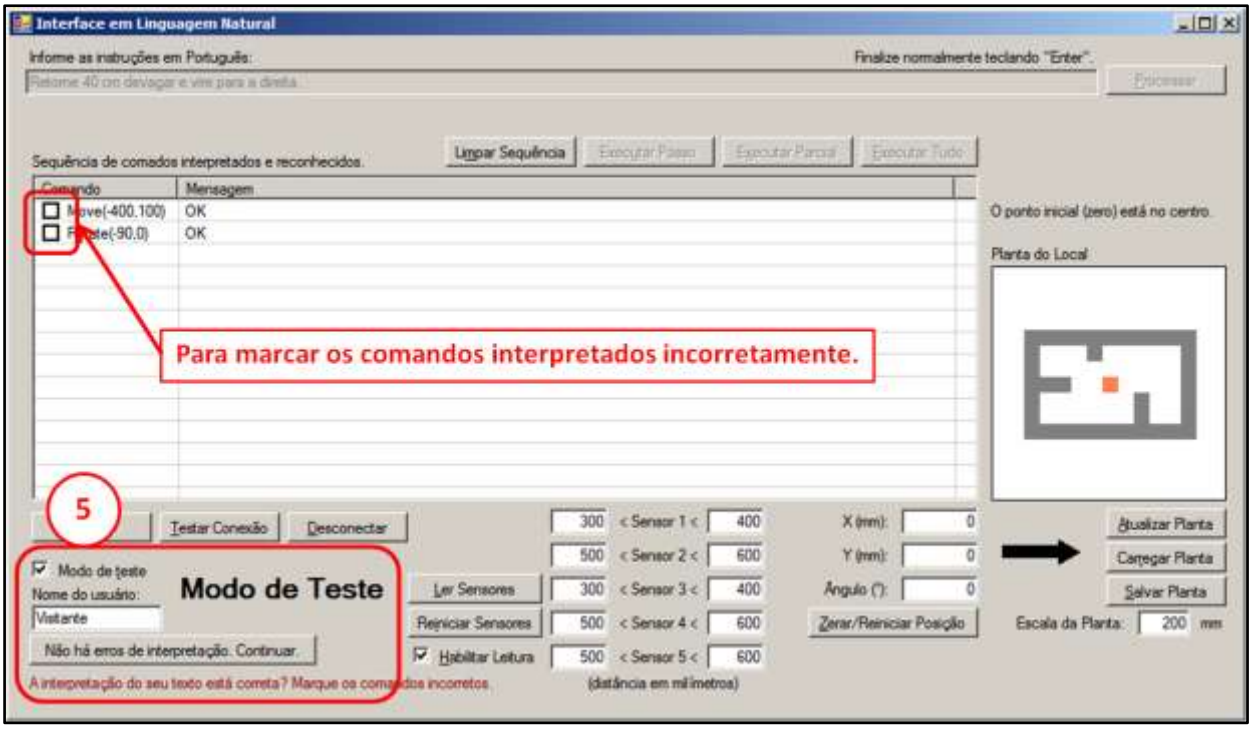


Figura 4.2.4 – Elementos de controle do sistema robótico em "Modo de Teste".

4.2.2 VISÃO GERAL DA FORMA DE UTILIZAÇÃO DA IFLN

O método de utilização da Interface em Linguagem Natural é intuitivo e simples, podendo ser descrito pela sequência de passos a seguir:

1) Teclar as instruções em Português;

2) Verificar se a sequência de comandos interpretados corresponde ao resultado esperado, sendo possível corrigir as instruções em Português, caso seja necessários; e

3) Executar a sequencia de comandos ou por inteiro, ou passo a passo ou parcialmente a partir de um ponto escolhido.

4.2.3 DETALHAMENTO DA FORMA DE UTILIZAÇÃO

No primeiro passo, para teclar em Português as instruções desejadas para que o robô execute, é importante observar as regras gramaticais de pontuação usualmente aplicável no contexto da oração assim como a ortografia. Pontuação incorreta, abreviações e palavras escritas incorretamente terão efeitos negativos sobre a análise do texto. Neste momento é importante observar que frases bem elaboradas, com a quantidade correta e suficiente de palavras, capazes de expressar claramente o que se propõe sem simplificações, produzem os melhores resultados. É recomendável evitar as redações que apresentam somente o núcleo da instrução, pois, nas orações nestes casos, não apresentam os recursos linguísticos que poderão auxiliar na interpretação do texto.

Por exemplo, a oração "Eu quero que o robô ande para frente agora" é preferível em relação à oração "Ande para frente", pois é simples e direta demais.

Isto se deve principalmente pelo uso de Sujeito Oculto, um recurso característico da Língua Portuguesa. Seu uso, associado à natural conjugação verbal, que por si só ocasiona a geração de várias palavras diferentes para um mesmo verbo, decorre em situações difíceis de serem contornadas com o processamento estatístico (estocástico) da linguagem natural. Ou seja, assumindo-se que uma frase possui como constituintes um Sujeito, um Verbo e um Predicado, não é possível reconstituir corretamente a presença de um elemento oculto na frase (o Sujeito) por meios puramente estatísticos, que depende da contagem de ocorrências em conjunto de elementos nos textos de treinamento. A devida solução para este que se constitui, por si só, em um problema linguístico, depende de *software* e abordagem apropriada (NETO, 2010).

Continuando, para indicar a conclusão das instruções em Português e submeter o texto para interpretação pelo sistema, basta teclar "*Enter*" (entra) ou então clicar no botão "Processar", como intuitivamente se pode imaginar. Outra forma de se indicar a conclusão das instruções é utilizar a tecla "*Tab*" (tabulação), uma forma bem comum de navegação em formulários de Sistemas Operacionais baseados em janelas. Portanto, são três formas de indicar a conclusão das instruções, uma vez que o Ponto Final indica apenas o final de uma oração e uma instrução pode conter mais de uma oração para estar completa.

Assim que a instrução em linguagem natural é interpretada, é apresentada a sequencia de comandos identificados a partir dela, para que o usuário tenha

oportunidade de verificar se está conforme o desejado inicialmente e empreender as adaptações ou correções necessárias no texto em Português.

Por exemplo, a frase “Eu quero que o robô ande para frente agora” será interpretada como o comando “*Move*” (mover). Porém, a frase “O canário da gaiola é amarelo” não resultará em nenhum comando reconhecido. Isto se deve ao fato de que o sistema, no seu contexto de utilização, que é o de interpretar instruções de movimento e navegação de um robô móvel, não está preparado para reconhecer instruções como a indicação de cor de um canário (ou outro objeto ou ser).

Está fora do escopo deste trabalho, porém, o sistema pode ser preparado para tal, caso esse tipo de instrução seja útil. Uma sugestão de desenvolvimento futuro seria a possibilidade de interpretar instruções de adição de referências da planta, associando-as a nomes, como por exemplo: “A cadeira está no quadrado 4,7” ou “O vaso é o ponto 3,6”.

Junto dos comandos, são apresentados os parâmetros característicos de cada um, bem como mensagens de erro e de aviso pertinentes ao caso de cada comando. Assim, em decorrência de ambos os exemplos do parágrafo anterior, na primeira frase faltou informar “quanto” ou “até onde” se deseja que o robô ande para frente bem como não foi informada a velocidade que seja o movimento, por exemplo. Portanto, serão apresentados dois avisos de que valores padronizados para distância e velocidade foram utilizados para completar o comando “*Move*” (mover). Se não for explicitado que o movimento desejado seja para frente, ou para trás, ou para direita, ou para a esquerda, igualmente, um aviso nos mesmos termos será apresentado. E para a segunda frase, será apresentado um erro de que não foi identificado ou reconhecido nenhum comando a partir dela. A razão disto está indicada no parágrafo anterior.

Verificando, então, a sequência de comandos, seus parâmetros, mensagens de erro ou aviso, o usuário poderá efetuar as correções no texto em Português como desejar, bastando posicionar o cursor no ponto do texto e reescrever o trecho, como seria feito em um editor de texto, por exemplo. É possível copiar, colar, inserir, apagar e substituir trechos ou o próprio texto por completo.

Uma vez que a sequência de comandos esteja satisfatória, o usuário terá a oportunidade de executá-la no robô. Neste caso, é possível controlar a forma como a execução será feita. Isto é útil para adaptar a execução às condições de uso ou

experimento, ou quando há falhas na comunicação sem fio com o robô, ou quando se deseja corrigir ou reposicionar o robô manualmente, por exemplo.

Para executar integralmente a sequência, de uma só vez, do início até o fim, sem parar, basta clicar no botão “Executar Tudo”. Seu atalho do teclado é “Alt+E”.

Para executar a sequencia a partir de um ponto escolhido da sequência até o final, basta clicar no ponto onde se deseja iniciar, na lista e comandos, e clicar no botão “Executar Parcial”. Seu atalho de teclado é “Alt+X”.

Para executar a sequência um passo de cada vez, a partir de um ponto escolhido, que pode ser o início ou não, em diante, basta clicar no botão “Executar Passo”. Seu atalho de teclado é “Alt+U”. A cada vez que este botão é clicado, ou seu atalho de teclado é usado, o passo atual é enviado ao robô e o passo seguinte é escolhido (para ser executado no próximo passo).

Um recurso adicional de execução, além dos relacionados acima, é clicar duas vezes com o cursor do *mouse* sobre qualquer comando da sequência, que executará somente este comando uma vez a cada clicada dupla. Este recurso é útil principalmente após as falhas de comunicação sem fio com o robô, que pode não receber devidamente um comando e faltar na execução de um passo.

Ao final, para introduzir novas instruções para serem interpretadas e recomeçar o ciclo, basta simplesmente teclar o novo texto em Português e seguir os passos desde o início, como apresentados acima, como se fosse a primeira vez.

4.2.4 VISÃO GERAL DO FUNCIONAMENTO

A Interface em Linguagem Natural se utiliza de recursos de *software* disponíveis no Sistema Robóticos para funcionar e também recursos auxiliares que são úteis no funcionamento da interface gráfica na qual se baseia.

Com relação aos recursos auxiliares, é possível citar:

- Rotinas para processamento dos eventos ocasionados pelos elementos;
- Funções de filtragem e processamento das informações apresentadas;
- Funções de conexão, ou comunicação, das informações apresentadas pelo usuário mediante a interface e as respostas do sistema ao usuário; e
- Funções responsáveis pelo comportamento dinâmico de elementos como botões, *feedback* visual e formatação de textos e números.

Estes últimos, que são recursos predominantemente visuais da interface, não constituem nenhuma parte crítica associada ao objetivo deste trabalho, sendo, na verdade, recurso secundário às questões que serão abordas a seguir. Portanto, eles serão apenas mencionados, mas não avaliados em detalhe.

Os recursos do Sistema Robótico que utiliza são as demais funções do bloco de Inteligência e Controle conforme descrito na seção 4.6 (Funcionamento do sistema robótico proposto).

4.3 O *SOFTWARE* DE INTELIGÊNCIA E CONTROLE

O *Software* de Inteligência e Controle é o conjunto de funções que desempenham o processamento crítico fundamental do Sistema Robótico proposto. É neste bloco funcional que ocorrem o processamento da linguagem natural, o sequenciamento dos comandos identificados e o controle do robô propriamente dito.

Independentemente da interface que seja utilizada para comunicação com o usuário, estas funções sempre serão utilizadas da mesma forma para realizar o processamento e o trabalho.

É composto de três grupos de funções:

1) Processamento da Linguagem Natural;
2) Lógica de Controle do Robô; e
3) Controle da Comunicação entre o Sistema e o Robô.

Cada um destes grupos de funções será abordado em mais detalhes nos parágrafos seguintes. Para uma visão geral dessas funções, ver Seção 4.1.

4.3.1 FUNÇÕES DE PROCESSAMENTO DA LINGUAGEM NATURAL

Estas são as funções que realizam o trabalho de transformar as instruções fornecidas pelo usuário em Português em uma sequência de comandos e parâmetros compatível com os recursos do robô e que podem ser efetivamente executados.

As funções de *software* que compõe este grupo são:

a) Separação de sentenças;
b) Separação de palavras ou "*tokens*";
c) Marcação das classes gramaticais ou sintáticas;
d) Avaliação de contexto das orações; e

e) Avaliação do significado e extração de informações do texto.

A) Separação de sentenças

Uma vez que as instruções que forem fornecidas pelo usuário em Português podem ser redigidas com de mais de um período, oração ou sentença, uma separação dessas se torna necessária. Isso significa individualizar os grupos de palavras que compõem cada período, do início até o ponto final, bem como encadear as frases na mesma ordem em que foram escritas.

Porém, não basta simplesmente considerar o "ponto final" como indicação de final de frase, pois as abreviaturas, assim como outros recursos de redação, também utilizam este sinal de marcação.

Este trabalho é realizado por método estocástico lançando mão de um Modelo de Entropia Máxima treinado com frases em Língua Portuguesa. Este método estocástico é capaz de indicar o final de uma frase considerando as palavras que a compõe e a probabilidade extraída do modelo treinado e atribuída em cada sequência.

No caso do *SharpNLP*, adotado como biblioteca de funções que realizam este trabalho, a função denomina-se "*SentenceDetect*". Sua entrada é uma *string* (cadeia de caracteres) com o texto digitado pelo usuário e sua saída é uma matriz de *strings*, sendo que cada elemento dessa matriz é uma cadeia que contém a sequência que possui a maior probabilidade de ser um período distinto, conforme o Modelo de Entropia Máxima.

B) Separação de palavras ou "*tokens*"

Para que seja possível fazer a análise sintática e atribuir funções gramáticas às palavras da oração é preciso que essas palavras sejam individualizadas corretamente. De maneira geral este não é um trabalho difícil de ser realizado em língua Portuguesa, uma vez que as palavras são geralmente separadas por espaços.

Uma vez que isto nem sempre ocorre, como no caso de contrações do tipo "d'água", onde é possível encontrar os componentes "d" e "água" e que realmente significam "de água", esses componentes são denominados *tokens*, porque são elementos que apresentam funções sintáticas capazes de proporcionar a ordem adequada à oração. Em outras palavras, e conforme o exemplo suscitado, "d" não é propriamente uma palavra, mas assume a mesma função sintática da palavra "de".

Além das palavras, os sinais de pontuação são importantes elementos de organização do período e precisam ser considerados para uma correta análise sintática. Eles também são *tokens*.

Cabe observar que, em se tratando de Língua Natural, há geralmente situações de ambiguidade, como as abreviaturas. Neste caso, palavras como "Senhor", "metro" e "esquerda" podem ser abreviados respectivamente como "Sr.", "m." e "esq.". Ou seja, no caso de "Sr.", não existe uma palavra "Sr" seguida da pontuação "ponto final", mas sim o conjunto todo deve ser reconhecido e tratado como um único *token*.

O trabalho de separação dos *tokens* é realizado também por método estocástico lançando mão de um Modelo de Entropia Máxima treinado com frases em língua Portuguesa. Da mesma forma, é capaz de indicar o ponto de separação mais provável entre os *tokens* com base nestes e na probabilidade extraída do modelo treinado, atribuída a cada sequência.

No caso do *SharpNLP*, adotado como biblioteca de funções que realizam este trabalho, a função denomina-se "*Tokenize*". Sua entrada é uma *string* (cadeia de caracteres) com o período (frase) individualizado e sua saída é uma matriz de *strings*, sendo que cada elemento dessa matriz é o que possui a maior probabilidade de ser um *token* distinto, conforme o Modelo de Entropia Máxima.

C) Marcação das classes gramaticais ou sintáticas

Com os *tokens* e as palavras individualizados e mantidos na mesma sequência em que foram escritos pelo usuário no texto em Português e em linguagem natural, é possível determinar suas classes gramáticas ou classes sintáticas.

Este trabalho, quando executado manualmente por um humano, é formalizado com regras gramaticais conhecidas. Mesmo que seja possível transcrever e executar essas regras de forma computacional, quando se trata de Linguagem Natural há geralmente tantas formas diferentes de exprimir uma mesma ideia em palavras que é bem fácil encontrar ambiguidade na aplicação dessas regras. Ou seja, simplesmente conhecer as regras não é suficiente para realizar a análise sintática e é preciso conhecer as diversas formas e possibilidades de uso e de construção de frases para ser possível resolver tais ambiguidades.

Por intermédio de métodos estocásticos e dos Modelos de Entropia Máxima é possível obter resultados teóricos bastante elevados, superiores aos 95% de acerto, conforme mostrado por Ratnaparkhi (1997).

É desta forma que se executa este processamento no Sistema Robótico proposto. Novamente aqui, o modelo é treinado com grande quantidade de frases em Língua Portuguesa, cujas palavras e *tokens* foram corretamente marcados conforme a gramática.

Deve-se, então, submeter a sequência à respectiva função do *SharpNLP*, cujo nome é "*PosTagger*". Sua entrada é por uma matriz de elementos *strings* (cadeia de caracteres), cada qual contendo o *token* que constitui a frase que se deseja obter as marcações sintáticas. E sua saída é uma matriz de *strings*, contendo em cada elemento a classe gramatical ou sintática de maior probabilidade, conforme o Modelo de Entropia Máxima, associada ao respectivo *token* em questão.

Para esta função, um aspecto digno de destaque é que os *tokens* são inseridos um por vez, fazendo-se, para cada um, uma chamada diferente na função. Para que o resultado seja mostrado correto, os *tokens* devem ser inseridos na mesma ordem em que se encontram no período conforme escrito pelo usuário. E para iniciar a análise de novo período, é necessário reiniciar (zerar) seus registradores internos (da função).

D) Avaliação de contexto das orações;

Uma vez dispondo da uma correta separação de palavras, *tokens* e sinais de pontuação, onde os problemas de ambiguidade sintática apresentam muito baixa probabilidade de se apresentarem, torna-se possível avaliar o contexto básico das orações que compõem os períodos redigidos pelo usuário.

Saber o contexto básico significa determinar qual dos contextos mapeados pelo Sistema Robótico é o mais provavelmente associado à oração. Isto é importante para que seja feito um processamento mais detalhado das palavras e *tokens*.

Neste ponto, o trabalho é feito por método determinístico, por intermédio de um léxico ou dicionário de palavras chaves conhecidas e a probabilidade associada a cada um dos contextos mapeados. A estratégia é a de "saco de palavras" ("*bag of words*"), ou seja, é considerada a presença da palavra e sua probabilidade associada a cada contexto, mas não sua ordem na estrutura.

Foi concebido e desenvolvido para o Sistema Robótico um algoritmo para este fim. Consiste na função "IdentificaCompandosEParâmetros" do *software* em C#. Sua entrada são duas matrizes de *strings*, sendo a primeira para os *tokens* e a segunda para as marcações sintáticas.

Esta função de *software* também concatena a execução de outras partes do algoritmo desenvolvido, que serão esclarecidas logo em seguida, coletando seus resultados para compor a saída final do algoritmo. Contudo, executa toda a avaliação de contexto antes de executar as demais partes.

A saída é uma matriz bidimensional de *strings*. A primeira coluna contém a *string* do comando a ser executado com seus parâmetros. A segunda coluna contém a *string* com as mensagens de aviso ou de erro decorrentes do processamento executado para determinar o comando a partir do texto em linguagem natural. Cada linha contém um comando a ser executado e na sequência crescente de ordinalidade. A saída será usada para inclusão na lista a ser exibida ao usuário para posterior execução.

No Sistema Robótico proposto, as orações que não indicam relação com os contextos mapeados produzem mensagem de erro reportando contexto desconhecido e são ignoradas. As orações que apresentam ambiguidade sobre qual contexto pertencem produzem mensagem erro ao usuário para reescrever o trecho de outra forma.

E) Avaliação do significado e extração de informações do texto;

Desde que se disponha das orações já separadas em seus *tokens* e dos contextos mapeados mais prováveis a que pertençam, é possível realizar uma análise mais detalhada do seu significado e estrutura para extrair as informações.

Esta análise mais detalhada para avaliação do significado e extração da informação é feita utilizando uma aplicação prática do conceito linguístico de Esquema de Casos ("*Case Frames*") proposto por Fillmore (1968a e 1968b). Ou seja, os contextos são mapeados inicialmente através dos seus núcleos verbais que apresentam ou necessitam de complementos verbais e nominais, conforme o Esquema de Caso em uso. Auxiliando esta análise existe um dicionário com as palavras apresentando seus significados vinculados dentro dos contextos (e Esquemas de Casos) mapeados previamente. Isto permite ao *software* reconhecer e reagir às estruturas da oração conforme apresentadas pelo usuário em linguagem natural.

Por meio de método determinístico, ou seja, através de trechos de *software* planejados e escritos para tal, inicialmente é determinado o conjunto de Esquemas de Casos da oração através do seu verbo, ou seja, o núcleo do Esquema de Caso. Ele então determinará as possíveis alternativas de estruturação das informações na frase, possibilitando aplicação de procedimentos para extração dos dados.

Neste momento, se algum dado que deveria estar presumivelmente presente não possa ser encontrado (se não foi escrito, se foi escrito de alguma forma não prevista ou se foi um erro de grafia), é gerada mensagem de alerta informando a ausência do dado. Mas um valor padrão é assumido em seu lugar, caso seja possível fazer isso dentro do contexto do significado. A mensagem de alerta é então completada para informar a adoção de valor padronizado.

Então, a saída resultante será um ou mais comandos para a lista a ser exibida ao usuário, com seus respectivos parâmetros de execução, sejam eles informados pelo próprio usuário ou presumidos por padronização.

Sendo continuação do algoritmo concebido e desenvolvido para o Sistema Robótico, consiste nas funções “ComandoMove”, “ComandoRotate” e “ComandoRotear” do *software* em C#. Suas entradas e saídas são semelhantes. A entrada são duas matrizes de *strings* (cadeia de caracteres), uma contendo a sequência de *tokens* da oração conforme escritos pelo usuário e previamente processados pelo sistema robótico (descrito anteriormente nos itens “a”, “b” e “c”) e outra contendo as respectivas marcações gramaticais ou sintáticas. Sua saída é uma matriz bidimensional de *strings* (cadeia de caracteres) para os comandos e mensagens de aviso/erro com a mesma estrutura da saída do item “f” desta seção.

4.3.2 FUNÇÕES DA LÓGICA DE CONTROLE DO ROBÔ

Estas são as funções que permitem a organização do controle do robô de acordo com as funções que ele dispõe, os comandos reconhecidos e a sequência de utilização tanto pelo usuário quanto pelo sistema.

Porém, antes de abordar a questão da lógica de controle do robô, será preciso conhecer primeiro as funções tanto do robô quanto do sistema para, em seguida, verificar como a lógica de controle funciona.

As funções do Robô e do Sistema Robótico

Respeitando os objetivos propostos e a simplicidade, as seguintes funções do robô foram mapeadas para reconhecimento e utilização:

a) Mover ("*move*" em inglês);

b) Girar ou rotacionar ("*rotate*" em inglês); e

c) Ler sensores.

Além das funções que o robô dispõe e que foram listadas acima, outras funções foram adicionadas ao sistema cujos comandos podem ser reconhecidos a partir do texto do usuário:

d) Rotear; e

e) Andar para o lado.

a) Função do robô: Mover

Refere-se à capacidade do robô se deslocar para frente ou para trás, a uma velocidade escolhida e por uma distância também escolhida. É acionada através do recurso de programação "Andar Reto" da seção A.1.8 (Programação – funções disponíveis).

b) Função do robô: Girar, virar ou rotacionar

Refere-se à capacidade do robô girar sobre sim mesmo para a direita ou para a esquerda, a uma velocidade escolhida e por um ângulo também escolhido. É acionada através do recurso de programação "Rotacionar" da seção A.1.8 (Programação – funções disponíveis).

c) Função do robô: Ler Sensores

Refere-se à capacidade do robô girar verificar a distância aproximada até o obstáculo através de seus sensores infravermelhos e transmitir o resultado ao computador hospedeiro. É acionada através do recurso de programação "*Status* dos sensores" da seção A.1.8 (Programação – funções disponíveis).

d) Função do sistema: Rotear

Refere-se a uma abstração natural das duas principais funções disponibilizadas pelo robô listadas acima: Mover e Rotacionar. Ou seja, com uma associação inteligente dos comandos Mover e Rotacionar é possível fazer o robô percorrer uma trajeto ou caminho previamente estabelecido ou calculado.

Neste caso, dotou-se o sistema de um recurso elementar de mapear o local aonde se localiza o robô e de formas de buscar o caminho entre pontos de origem (onde está o robô) e destino (aonde ele deve chegar). Uma vez dispondo do caminho, ele é decomposto de forma que possa ser descrito pelas funções Mover e Rotacionar.

Os procedimentos computacionais e rotinas de processamento foram concebidos e elaborados para o sistema robótico proposto.

e) Função do sistema: Andar para o lado

Refere-se a outra abstração natural das duas principais funções disponibilizadas pelo robô (Mover e Rotacionar), pois é bastante prático e útil indicar comandos do tipo “fique 30 cm mais para a esquerda”. Ou seja, além de se deslocar para frente ou para trás, pode ser previsivelmente necessário um deslocamento para a esquerda ou para direita.

Neste caso, em decorrência da configuração das rodas do robô (ver Apêndice A.1), requer uma rotação. Esta composição de movimentos (rotacionar e mover) pode não ser necessária em alguns tipos de robôs capazes de se deslocar lateralmente. Contudo, é o caso do robô Curumim adotado para realização desta dissertação. É interessante notar que esta composição de movimentos coincide com solução dos movimentos humanos, pois preferimos nos deslocar para frente, sendo que é desconfortável e pouco usual o andar de lado por distâncias maiores que um passo.

A lógica de controle propriamente dita

Para que o resultado esperado seja obtido, as funções do robô são executadas de forma coordenada, ou seja, na sequencia correta de comandos de acordo com a interpretação das instruções do usuário como também seu uso lógico conforme o próprio comando.

É assim, então, que o sistema deve executar:

- Verificação dos sensores quando a existência ou não de obstáculos no caminho a ser seguido antes de enviar um comando de movimento ao robô;
- Produção de uma lista de comandos de movimento e rotação na ordem apropriada para que o robô percorra a rota identificada para atingir um dado ponto de destino; e
- Produção da sequência apropriada de rotação e movimento para permitir que o robô execute um deslocamento para os lados (direito e esquerdo) quando assim for instruído pelo usuário.

4.3.3 FUNÇÕES DE CONTROLE DA COMUNICAÇÃO ENTRE O SISTEMA E O ROBÔ

Estas são as funções que permitem ao Sistema Robótico proposto integrar ou não o robô na execução dos comandos conforme instrução do usuário, bem como processar o resultado da execução.

A seção 5.3 (Algoritmos implementados) apresenta mais detalhes.

4.4 A INTERFACE COM O ROBÔ

A Interface com o Robô se constitui de duas partes intimamente interdependentes, sendo uma *software* ("*driver*") e outra *hardware* (transceptor).

A função da parte *software* é abstrair as especificidades de controle dos recursos de comunicação do *hardware*, sendo disponibilizada pelo fabricante através de uma biblioteca de funções e um programa instalador. Neste caso, o instalador realiza a configuração que é necessário ao sistema operacional do computador enquanto que a biblioteca de funções é vinculada pelo *software* do próprio Sistema Robótico.

Sua implementação no Sistema Robótico decorre basicamente de leitura, interpretação e execução do manual técnico do fabricante. Não será detalhada nesta dissertação. Mais alguns informações poderão ser encontradas no Apêndice A.1.

4.5 O ROBÔ

O robô utilizado no desenvolvimento do Sistema Robótico proposto é o Curumim, robô nacional para propósitos educacionais e de pesquisa. O Apêndice A.1 o apresenta com mais detalhes.

Foram escolhidas partes, e não a totalidade, dos recursos disponibilizados por ele para concepção do Sistema Robótico. A seção 5.1 (Planejamento e restrição de contexto) esclarece mais sobre este assunto.

4.6 FUNCIONAMENTO DO SISTEMA ROBÓTICO PROPOSTO

Uma vez conhecidos os blocos funcionais, e as funções propriamente ditas, que compõem o *Software* de Inteligência e Controle, será possível aprofundar mais no funcionamento do Sistema Robótico proposto.

A seguir será feita análise de como essas funções são utilizadas. Os recursos deste bloco estão intrinsecamente relacionados com a proposta deste trabalho. Seus componentes são:

A. Funções estocásticas de separação de sentenças;
B. Funções estocásticas de separação de palavras ou "*tokens*";
C. Funções estocásticas de marcação das classes gramaticais ou sintáticas;
D. Funções determinísticas de avaliação de contexto das orações;
E. Funções determinísticas de avaliação do significado e extração de informações do texto;
F. Procedimentos de lógica e controle do robô;
G. Procedimentos de controle da comunicação entre o sistema e o robô.

Durante a operação, o sistema realiza o processamento das instruções em linguagem natural para obtenção da sequência de comandos a serem executados. A figura 4.6.1 ilustra a maneira geral como o Processamento da Linguagem Natural é utilizado pelo Sistema Robótico.

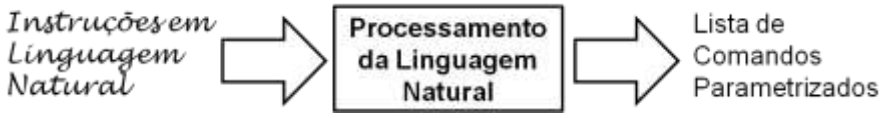


Figura 4.6.1 – Visão geral do processamento das instruções em linguagem natural.

O processamento ocorre da seguinte forma:

1) Análise sintática das orações contidas na instrução;
2) Análise semântica e pragmática das orações;
3) Identificação dos comandos reconhecidos; e
4) Mapeamento da sequência de comandos na lista de execução.

A análise sintática é executada por método estocástico, da seguinte forma:

1) O texto em linguagem natural é submetido ao processamento por um modelo de Entropia Máxima que foi treinado para realizar a identificação e separação de cada oração ou período que compõem a instrução (A – Funções estocásticas de separação de sentenças);
2) Cada uma dessas orações é então submetida a outro Modelo de Entropia Máxima, porém treinado para identificar e separar as palavras que compõem a oração em análise (B – Funções estocásticas de separação de sentenças); e
3) A sequência de palavras que foram identificadas e que compõem a oração em análise deve ser submetida, então, a um terceiro Modelo de Entropia Máxima, desta vez um que foi treinado para identificar as funções sintáticas das palavras ao longo da oração (C – Funções estocásticas de marcação das classes gramaticais ou sintáticas).

As funções estocásticas baseadas nos Modelos de Entropia Máxima encontram-se disponíveis numa biblioteca de domínio público denominada *SharpNLP*. Essas funções foram vinculadas via *Visual Studio*® ao projeto do Sistema Robótico, tornando-se disponíveis para utilização no ambiente de programação.

Contudo, as funções estocásticas dependem de Modelos de Entropia Máxima que foram treinados especificamente para a aplicação que se deu durante este trabalho.

As análises semântica e pragmática são realizadas juntas, de uma só vez, por método determinístico, ou seja, por conjunto de parâmetros e procedimentos de

software, previamente preparados para esta finalidade. O objetivo desta análise é extrair o significado contido nas orações. A sequência em que este processamento ocorre é a seguinte:

1) A cada palavra e sua respectiva função sintática é atribuído um valor extraído de um dicionário que indicará a probabilidade dela estar relacionada aos contextos previamente mapeados pelo sistema (D – Funções determinísticas de avaliação de contexto);
2) A somatória das probabilidades da sequência de palavras, e funções sintáticas, indicam a probabilidade de estarem relacionadas a cada contexto mapeado pelo sistema; e
3) O contexto de maior probabilidade é então sinalizado como sendo o relacionado àquele conjunto de palavras.

Uma vez determinado o contexto do significado mais provável para aquele conjunto de palavras, elas são submetidas ao procedimento que irá extrair os parâmetros e dados ali contidos (E – Funções determinísticas de avaliação do significado e extração de informações do texto) levando em conta o contexto a que pertencem. Irá também identificar os erros e parâmetros faltantes, para substituí-los por valores padronizados (pré-estabelecidos) pelo sistema de forma que o comando a ser inserido na lista tenha condições de ser útil.

Isto é feito com auxílio de um léxico ou dicionário (recurso heurístico) que associa valores e parâmetros úteis (computáveis) para cada palavra, dentro dos diferentes contextos mapeados pelo sistema e de um procedimento de *software* que se encarrega de realizar a verificação de consistência do resultado, para gerar as mensagens de erro e avisos conforme o caso.

Os contextos mapeados previamente pelo sistema referem-se às funções possíveis de serem executadas pelo robô conectado ao sistema (mover, girar, fotografar, pinçar etc.) e os serviços decorrentes do uso inteligente dessas funções (navegar até um lugar, mostrar um objeto, correr até um lugar, posicionar-se em relação a um lugar etc.). O mapeamento é feito, com sua respectiva vinculação às palavras associadas, por intermédio da teoria dos Esquemas de Casos ("*Case Frames*", em inglês). O Capítulo 3 esclarece sobre esta teoria.

O conjunto de palavras (oração) que apresentar significado que não puder ser mapeado a nenhum contexto reconhecido pelo sistema será considerado como sem significado útil e produzirá uma mensagem de aviso ao usuário, indicando o fato.

O resultado deste procedimento de *software* é a identificação de cada comando constante na instrução do usuário, seus parâmetros e as mensagens de erro e aviso, conforme o caso.

Este processo é executado ciclicamente até que todo o conjunto de orações seja analisado. A sequência em que as orações forem dispostas indicará a sequência de comandos resultante na caixa de comandos apresentada ao usuário, na interface em linguagem natural.

A figura 4.6.2 apresenta a organização do bloco de funções responsáveis pelo Processamento da Linguagem Natural do Sistema Robótico proposto.

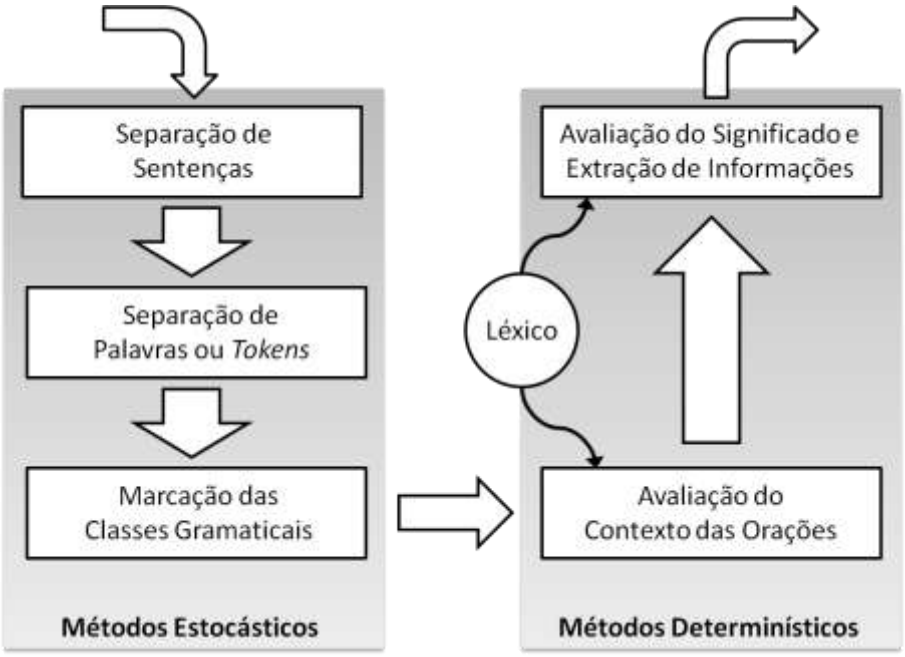


Figura 4.6.2 – Organização das funções de Processamento da Linguagem Natural.

Uma vez dispondo da sequência de comandos, com seus respectivos parâmetros, o usuário poderá solicitar a execução pelo Sistema Robótico e, consequentemente, pelo Robô. A execução pode ser controlada conforme indicado na seção 4.3.2 deste trabalho. Neste caso, são acionados os "F – Procedimentos de lógica e

controle do Robô", utilizando os comandos e parâmetros previamente organizados na lista sequencial mencionada anteriormente. A figura 4.6.3 ilustra a maneira geral como a execução e o controle dos comandos são realizados pelo Sistema Robótico.

Figura 4.6.3 – Visão geral da execução e do controle dos comandos.

Para que sejam efetivamente executados pelo Robô, o Procedimento de lógica e controle faz uso dos "G - Procedimentos de comunicação entre o robô e o sistema de controle". As questões de envio e recebimento de comandos e parâmetros do sistema para o robô, acompanhamento da execução pelo robô, recebimento da leitura dos sensores do robô para o sistema e os problemas que podem ocorrer durante a comunicação, com as mensagens de erro relativas, são processados neste ponto do Sistema Robótico.

A figura 4.6.4 representa a organização do bloco de funções responsáveis pela Lógica de Controle e Comunicação do Sistema Robótico proposto.

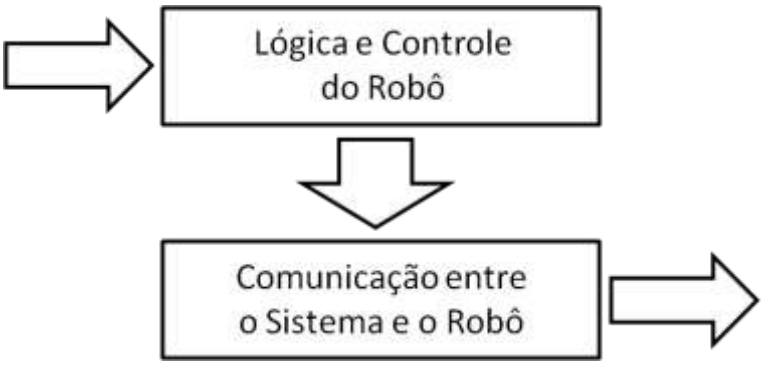


Figura 4.6.4 – Organização das funções de Lógica de Controle e Comunicação.

Foi mostrada acima uma visão geral dos recursos responsáveis pelo funcionamento do Sistema Robótico que permite os resultados objetivados por esta

dissertação e assim possibilitar a compreensão do relacionamento entre eles. A Seção 5.3 (Algoritmos e procedimentos implementados) deste Capítulo mostrará com mais detalhes o funcionamento desses recursos.

4.7 CONSIDERAÇÕES FINAIS

Foi mostrado que o sistema robótico proposto é composto de uma parte física (o *hardware*), tais sejam o computador, os dispositivos de comunicação e o robô propriamente dito, e uma parte lógica (o *software*), tais sejam, os recursos de inteligência artificial e processamento da linguagem natural, os controladores dos dispositivos de comunicação e as bibliotecas de funções de programação tanto destes dispositivos como também do robô.

Como forma de organizar o *software*, as partes do sistema foram agrupadas conforme a sua funcionalidade, ou seja:

- Interface com o usuário;
- Processamento da Linguagem Natural;
- Lógica de Controle do Robô; e
- Controle da Comunicação entre o Sistema e o Robô.

A utilidade de cada parte do sistema foi esclarecida como forma de compreender como se relacionam entre si. Assim, é possível executar verificações de erros ou então expansão com novas funcionalidades, à medida da necessidade.

Foi mostrado o funcionamento da Interface em Linguagem Natural, seus elementos visuais, suas propriedades e a forma possível de interação com o usuário.

Internamento, os elementos que permitem o funcionamento do *Software* de Inteligência e Controle foram apresentados, de forma a esclarecer sobre como o sistema realiza seu funcionamento.

Conhecer o robô Curumim e seus recursos, tanto físicos como de programação, é importante para compreender melhor este trabalho. A partir dele foram propostos experimentos para simulação de casos práticos de uso bem como foi possível elencar alguns recursos de linguagem natural para proposição das funcionalidades de *software* que se tornam necessárias para a devida extração de significado e dados.

No próximo capítulo, para orientar o desenvolvimento e o detalhamento dos recursos internos, serão apresentadas considerações sobre restrições e sobre o planejamento.

Nele também serão detalhados os algoritmos relativos a este trabalho em seu funcionamento, para concluir a descrição do Sistema Robótico proposto.

5. O DESENVOLVIMENTO DO SISTEMA ROBÓTICO

Será mostrado neste capítulo o desenvolvimento do Sistema Robótico proposto com mais detalhes, quer seja a composição do *software*, quer seja o trabalho necessário para sua realização, quer sejam as dificuldades encontradas, as soluções propostas e as decisões necessárias ao longo do caminho.

5.1 PLANEJAMENTO E RESTRIÇÃO DE CONTEXTO

Como premissas básicas para o Sistema Robótico proposto, seriam utilizados um robô brasileiro (o Curumim), a linguagem de programação C# (já que é moderna, orientada a objetos, dispõe de uma vasta biblioteca de funções e recursos denominada *dotNet* e possui suporte tanto da sua empresa criadora, a *Microsoft*, quanto da comunidade de *software* livre – Mono), peças de *softwares* e algoritmos de programação aplicados ao processamento da linguagem natural desenvolvidos pela comunidade de *software* livre (sendo possível utilizá-los sem pagamento de *royalities*) e a linguagem natural oficial do Brasil (o Português).

Porém, por razões de simplicidade, praticidade e limitação de tempo disponível para pesquisa de referencial teórico assim como concepção, desenvolvimento, testes e aprovação do sistema robótico proposto, algumas restrições foram estabelecidas.

Com relação ao *software*, as peças que foram consideradas para utilização (e consequente integração ao sistema robótico) assim como o resultado deste trabalho em si, precisariam ser livres de cobrança monetária de direitos autorais bem como o código ser aberto para consulta, alteração e reutilização. Em todos os casos, as menções e créditos autorais serão respeitados conforme as convenções e normas estabelecidas. Por conseguinte, serão desconsideradas soluções comerciais, que signifiquem cobrança de valores monetários sob qualquer título, ou outras peças de *software* que não permitam a consulta, a alteração ou a reutilização de seus códigos constituintes.

A Interface em Linguagem Natural, responsável pela interação com o usuário, seria implementada somente via *Windows Forms*, ou seja, janela de *software* tipo

formulário característica do sistema operacional utilizado. Não serão consideradas para avaliação nesta dissertação outras variantes possíveis de interface, senão para algum comentário útil à clareza do texto.

Com relação ao *hardware*, o sistema robótico proposto será desenvolvido sobre computador pessoal tipo PC com sistema operacional *Windows*. Estará conectado a um único robô, que neste caso será o brasileiro Curumim.

Em decorrência, o planejamento das suas funções e serviços possíveis está fortemente relacionado aos recursos disponíveis no robô. Ainda assim, não será a totalidade das funcionalidades que serão utilizadas, mas tão somente as que se mostraram mais interessantes aos propósitos deste trabalho.

Dentre as funcionalidades e recursos presentes no Curumim, foram consideradas somente:

- Funções de atuação: Deslocamento e Rotação; e
- Funções de sensoreamento: detecção de obstáculo por infravermelho.

O robô Curumim herdou o paradigma de movimentação humano, ou seja, desloca-se para frente ou para trás. Como não possui suporte para deslocamento para os lados, é necessário executar uma rotação quando este tipo de deslocamento é necessário.

Os sensores de obstáculo funcionamento por reflexão de sinal infravermelho. O conceito de funcionamento é sinalizar a presença ou não de obstáculo dentro de certa distância informada, ao invés de retornar a distância propriamente dita até o obstáculo.

Não foram consideradas outras funcionalidades e recursos disponibilizados pelo robô, tais como:

- Funções de atuação: Pinçar e Fazer Curva; e
- Funções de sensoreamento: câmera de vídeo.

Uma restrição imposta em decorrência de rápidos testes empíricos é a distância máxima possível de ser percorrida pelo robô durante a experimentação, transcrita do raio de alcance que permitisse comunicação sem fio confiável entre a Estação Radiobase e o Robô. Isto por que o ambiente de experimentação é bastante ruidoso, com várias fontes de comunicação sem fio como terminais *Wifi* (rede local sem fio), terminais *Bluetooth* (rede pessoal sem fio), telefones sem fio e dispositivos de alarme nos andares subjacentes. Outros ambientes testados, mais silenciosos, porém

não convenientes ao desenvolvimento, apresentaram em melhor alcance. O raio de alcance foi limitado em 5 metros em torno da Estação Radiobase.

5.2 PLANEJAMENTO E ESQUEMA DE CASOS

As funções de atuação disponíveis no robô Curumim dentro do contexto deste trabalho são o Deslocamento e a Rotação. Com elas, foram propostos três Esquema de Casos ("*Case Frames*") para serem implementados no *software* e, assim, permitir o reconhecimento de comandos apresentados por usuários através de instruções em Linguagem Natural em Português.

Os seguintes Esquemas de Casos foram planejados para o sistema:

a) Caso do Comando Mover;

b) Caso do Comando Rotacionar; e

c) Caso do Comando Rotear.

Os Esquemas de Casos são identificados pelos verbos de ação colocados na instrução pelo usuário. Cada Esquema de Caso necessita de seus papéis semânticos para estarem completos, que serão interpretados nos parâmetros do comando associado. Esses papéis semânticos são apresentados com o uso de outras palavras presentes nas orações. A tabela 5.2.1 exibe esses Casos e seus Papéis, sendo que o conjunto de símbolos "999" representa uma quantidade numérica.

Tabela 5.2.1 – Visão geral dos Esquemas de Casos e seus Papeis.

Caso	Papel
Mover	**Sentido**: frente, trás, esquerda, direita. **Distância**: 999 **Unidade** (m, cm, mm, quadrados). **Velocidade**: lento, normal, rápido.
Rotacionar	**Sentido**: direita, esquerda, trás.
Rotear	**Destino**: nome da referência. **Velocidade**: lento, normal, rápido.

5.2.1 ABSTRAÇÕES AGREGAM UTILIDADE

É oportuno observar que o robô Curumim não dispõe de recursos apropriados para deslocamento para os lados, mas tão somente para frente e para trás. Assim, o comando Mover intuitivamente previsto por um usuário humano como sendo

possível também para os lados requer uma associação dos comandos Rotacionar antes do Mover. Isto pode ser determinado de forma lógica avaliando a combinação de palavras usadas pelo usuário na instrução em linguagem natural. Por exemplo, observando as frases abaixo, é possível notar que os verbos de ação são “ficar” e “ir”, encontrados conjugados, são usados com as palavras “esquerda” e “direita”.

- Eu quero que você fique 20 cm mais para a esquerda.
- O robô deve ir 1 metro para a direita.

Assim, uma regra lógica conveniente para estes casos é verificar a presença de palavras que indicam sentidos que não o reto (para frente ou para trás) e acrescentar um comando Rotacionar antes do comando Mover.

Intuitivamente, um usuário humano leigo ou inexperiente com robôs, não se preocupará com questões como as características do projeto ou dos recursos de mobilidade do robô, sendo mais conveniente que o sistema robótico seja capaz de resolver o problema de reconhecer o que o usuário seja e executar devidamente.

Não se trata de utilização direta de uma única função do robô. O resultado depende de avaliação de mais características da frase para compor uma combinação ordenada de duas funções. Portanto, mesmo que necessária em decorrência das características do robô Curumim, este caso requer uma pequena abstração no uso dos comandos.

Prosseguindo com o mesmo princípio, um dos Esquemas de Casos não decorre diretamente de funções do robô, mas se trata de uma abstração mais elaborada: o caso Rotear. Ou seja, através do encadeamento de comandos Mover e Rotacionar, baseado num algoritmo de busca heurística em uma planta, previamente armazenada, do local onde se locomove o robô, é possível experimentar instruções mais simples ao operador humano, pois permite utilizar referências de lugares por nomes convencionais, do tipo:

- Acho melhor seguir até a “porta” e virar para a esquerda.
- Ele quer que você fique perto da “cadeira”.
- Corra para a “saída”.

Nos exemplos de frases acima, as palavras “porta”, “cadeira” e “saída” são pontos de referência reconhecíveis na planta do local.

Esta abstração adiciona um recurso inteligente ao sistema, permitido que execute um serviço útil, tanto no ponto de vista da atuação do robô quanto da interpretação da comunicação do usuário humano.

5.2.2 PLANEJAMENTO DO LÉXICO

Para identificar os Esquemas de Casos previstos, verbos de ação precisam ser incluídos no Léxico que o Sistema Robótico Proposto utiliza. Uma decorrência natural da língua portuguesa é a flexão de verbos de acordo com o tempo e a pessoa, o que dá origem a várias palavras para um mesmo verbo, que também precisam ser incluídas no Léxico.

São mostrados na tabela 5.2.2 exemplos de verbos que precisarão ser considerados no mapeamento dos Esquemas de Casos previstos, já que são palavras que podem ser interpretadas em instruções de usuários.

Tabela 5.2.2 – Exemplo de verbos de ação úteis para o léxico.

Verbo	**Variantes também incluídos no léxico**
Andar	Andasse, Ande, Andando, Andado
Avançar	Avançasse, Avance, Avançando, Avançado
Correr	Corresse, Corra, Correndo, Corrido
Deslocar	Deslocasse, Desloque, Deslocando, Deslocado
Direcionar	Direcionasse, Direcione, Direcionando, Direcionado
Esperar	Esperasse, Espere, Esperando, Esperado
Estar	Estivesse, Esteja, Estando, Estado
Ficar	Ficasse, Fique, Ficando, Ficado
Fugir	Fugisse, Fuja, Fugindo, Fugido
Girar	Girasse, Gire, Girando, Girado
Ir	Fosse, Vá, Indo, Ido
Mexer	Mexesse, Mexa, Mexendo, Mexido
Mover	Movesse, Mova, Movendo, Movido
Orientar	Orientasse, Oriente, Orientando, Orientado
Olhar	Olhasse, Olhe, Olhando, Olhado
Recuar	Recuasse, Recue, Recuando, Recuado
Retornar	Retornasse, Retorne, Retornando, Retornado
Seguir	Seguisse, Siga, Seguindo, Seguido
Voltar	Voltasse, Volte, Voltando, Voltado
Virar	Virasse, Vire, Virando, Virado

Para identificar os Papéis Semânticos dos Esquemas de Casos previstos, outras palavras precisam ser incluídas no Léxico do Sistema Robótico. Essas palavras

podem representar em si mesmas o Papel Semântico que será interpretado no parâmetro que completará o comando em questão ou auxiliarão na sua identificação. Na tabela 5.2.3 são mostrados alguns exemplos de palavras a serem incluídas no Léxico para identificar os Papéis Semânticos previstos e que podem ser interpretados à partir da instrução do usuário.

Tabela 5.2.3 – Exemplo de outras palavras úteis para o léxico.

Papel Semântico	**Palavra**
Sentido	Direita, Direito, Direto, Frente, Reto, Retaguarda, Trás, Traseira
Velocidade	Apressado, Calmo, Devagar, Lento, Ligeiro, Vagarosamente
Distância	Metro, Metro**s**, m, cm, centímetro, milímetros

Assim, palavras como “Frente” e “Retaguarda” indicam cada uma um sentido em si. Ao serem utilizadas em instruções para o robô Curumim, podem ser interpretadas pelos Casos Rotacionar ou Mover, dependendo do verbo que for utilizado. Por exemplo:

- Você precisa prosseguir em frente… → Caso Mover
- Eu quero que se volte para a retaguarda. → Caso Rotacionar

Ou então, palavras como “centímetro” e “devagar” sugerem e qualificam a presença de quantidade numérica, que poderão completar o Papel Semântico de Distância e Velocidade no Esquema de Caso Mover, por exemplo.

5.3 ALGORITMOS E PROCEDIMENTOS IMPLEMENTADOS

A seguir, serão apresentados e comentados os procedimentos e algoritmos implementados para composição do sistema robótico proposto.

Relembrando a proposta inicial desta dissertação, alguns são preexistentes e presentes em bibliotecas de funções disponíveis através na comunidade de *software* livre enquanto que outros foram concebidos e desenvolvidos especialmente para a necessidade da aplicação, com vistas aos resultados desejados. Serão detalhados somente os algoritmos novos que foram propostos.

A tabela 5.3.1 mostra onde essas peças de *software* estão localizadas na organização do sistema.

Tabela 5.3.1 – Blocos funcionais, Algoritmos e Procedimentos.

Bloco Funcional	Algoritmo (A) ou Procedimento (P)
Processamento da Linguagem Natural	(P) “Processa Linguagem Natural”
	(A) “Separação de Sentenças”
	(A) “*Tokenização* de Sentença”
	(A) “Marcação da Função Sintática”
	(A) “Separação de Orações pela Pontuação”
	(A) “Identificação de Comandos e Parâmetros”
	(A) “Rumar Para”
Lógica e Controle do Robô	(A) “Adição à Lista de Comandos”
	(P) “Executar Tudo”
	(P) “Executar Parcial”
	(P) “Executar Passo”
Comunicação entre o Sistema e o Robô	(P) “Executar Comando”
	(P) “Enviar Comando”

5.3.1 PROCEDIMENTO “PROCESSA LINGUAGEM NATURAL”

Após o usuário informar ao sistema o texto com as instruções que deseja executar, uma sequência de eventos rotineiros de processamento deve ser disparada e coordenada objetivando apresentar como saída a sequência de comandos, seus parâmetros e mensagens de alerta ou erro. Esta é a função deste procedimento de *software*.

Sua entrada é uma variável tipo *string* (cadeia de caracteres) contendo o texto fornecido pelo usuário. Não possui uma saída formal, porém dispara outros procedimentos e algoritmos que modificam os recursos do sistema.

Para ser desenvolvida nova Interface em Linguagem Natural (IFLN) com o usuário, outro procedimento como este deverá ser previsto, pois cada IFLN afetará elementos apropriados do sistema, que necessitarão de coordenação específica. A que foi implementada no sistema proposto, na versão atual, foi direcionada para os recursos do “*Windows Forms*”, ou seja, janela tipo formulário do sistema operacional.

A tabela 5.3.2 apresenta o procedimento que processa a linguagem natural.

Tabela 5.3.2 – Procedimento "Processa Linguagem Natural".

Passo	Descrição
01	Limpar a Lista de Comandos.
02	Converter o texto do usuário em minúsculas e separar em várias sentenças.
03	Para cada sentença do texto, executar:
04	Separar as palavras e *tokens*.
05	Marcar a função sintática de cada palavra ou *token*.
05	Se houver pontuação vírgula, ponto e vírgula ou "e" na sentença, executar:
06	Separar em várias orações pela pontuação.
07	Identificar os Comandos e Parâmetros nas orações.
08	Para cada Comando e Parâmetro identificado, executar:
09	Adicionar Comando, seus Parâmetros e Avisos/Erros na Lista de Comandos.
10	Fim do Para
11	Fim do Se
12	Identificar os Comandos e Parâmetros nas sentenças.
13	Para cada Comando e Parâmetro identificado, executar:
14	Adicionar Comando, seus Parâmetros e Avisos/Erros na Lista de Comandos.
15	Fim do Para
16	Fim do Para

5.3.2 ALGORITMO "SEPARAÇÃO DE SENTENÇAS"

A separação de sentenças é executada pela função "*SentenceDetect*" disponível na biblioteca do *SharpNLP*, que utiliza método estocástico baseada em um modelo de entropia máxima treinado em Português.

Sua entrada é uma variável tipo *string* (cadeia de caracteres) contendo a totalidade do texto informado pelo usuário.

Sua saída é uma matriz unidimensional de *strings* sendo que cada elemento contém as sentenças individualizadas da forma mais provável conforme o modelo treinado.

*5.3.3 ALGORITMO "*TOKENIZAÇÃO *DE SENTENÇA"*

A separação de palavras e *tokens* de cada sentença é executada pela função "*Tokenize*" disponível na biblioteca do *SharpNLP*, que utiliza método estocástico baseada em um modelo de entropia máxima treinado em Português.

Sua entrada uma variável tipo *strings* (cadeia de caracteres) contendo uma sentença.

Sua saída é uma matriz unidimensional de *strings* sendo que cada elemento contém as palavras ou os *tokens* individualizados da forma mais provável conforme o modelo treinado.

5.3.4 ALGORITMO "MARCAÇÃO DA FUNÇÃO SINTÁTICA"

A marcação de palavras e *tokens* de cada oração com sua correspondente função sintática é executada pela função "*PosTagger*" disponível na biblioteca do *SharpNLP*, que utiliza método estocástico baseada em um modelo de entropia máxima treinado em Português.

Sua entrada é uma matriz unidimensional de *strings* (cadeia de caracteres) contendo em cada elemento uma palavra ou *token* e organizada na mesma sequência em que aparece na oração.

Sua saída é uma matriz unidimensional de *strings* com a mesma dimensão da matriz de palavras/*tokens*, contendo em cada elemento a marcação da função sintática ("*Tags*") mais provável conforme o modelo treinado, respectivamente aos elementos da matriz de palavras/*tokens*.

5.3.5 ALGORITMO "SEPARAÇÃO DE ORAÇÕES PELA PONTUAÇÃO"

O texto do usuário pode ser escrito contendo, em uma mesma sentença ou período, mais de uma oração com conteúdo significativo para expressar comandos e parâmetros que precisarão ser identificados. De acordo com as regras gramaticais, essas orações podem ser separadas por sinais de pontuação e a palavra "e".

Sua entrada é uma matriz unidimensional de *strings* (cadeia de caracteres) contendo em cada elemento uma palavra ou *token* e organizada na mesma sequência em que aparece na sentença.

Sua saída consiste na composição de duas matrizes unidimensionais de *strings* cujos elementos são a sequência de palavras e *tokens* que compõem a oração, na primeira, e a sequência de marcações sintáticas ("*Tags*") respectivamente à palavras/*tokens*, na segunda.

Uma sugestão de desenvolvimento futuro é o estudo de aplicação de recursos para identificar mudanças na ordem sequencial das instruções, como a presença de expressões como "antes de" e "depois de", por exemplo.

A tabela 5.3.3 descreve o algoritmo "Separação de Orações pela Pontuação".

Tabela 5.3.3 – Algoritmo "Separação de Orações pela Pontuação".

Passo	Descrição
01	Atribuir à variável de elemento inicial da oração o valor zero.
02	Para elemento da matriz palavras/*tokens*, executar:
03	Se for pontuação (vírgula, ponto e vírgula ou "e"), executar:
04	Separar a oração do elemento inicial da oração até o elemento atual.
05	Acrescentar a oração à matriz de saída.
05	Separar as respectivas marcações de funções sintáticas da mesma forma.
06	Acrescentar as marcações à matriz de saída.
07	Atribuir à variável de ponto inicial da oração o valor do elemento atual.
08	Fim do Se
09	Fim do Para

5.3.6 ALGORITMO "IDENTIFICAÇÃO DE COMANDOS E PARÂMETROS"

Uma vez realizado o processamento sintático do texto fornecido pelo usuário, será preciso realizar a interpretação para identificar os comandos presentes e se foram ou não fornecidos todos os parâmetros de configuração necessários.

Este algoritmo realiza processamento semântico e pragmático simultâneos, em decorrência dos Esquemas de Casos previamente planejados para o sistema.

Sua entrada são duas matrizes unidimensionais de *strings* (cadeia de caracteres). A primeira matriz contém em cada elemento as palavras e *tokens* na sequencia em que foram digitados pelo usuário. A segunda matriz contém em cada elemento as marcações sintáticas ("*Tags*"), respectivamente a cada palavra ou *token* e na mesma ordem.

Sua saída é uma matriz bidimensional de *strings*. A primeira coluna será para os comandos identificados, já com seus respectivos parâmetros de configuração, organizados na mesma sequência em que aparecem no texto fornecido pelo usuário. A segunda coluna será para as mensagens de aviso ou erro identificadas pelo sistema durante o processamento, respectivamente a cada comando identificado.

Para executar a identificação dos comandos e parâmetros estruturou-se a solução para que fosse realizada em duas etapas, dividindo o problema da mesma forma:

1) Avaliação do Contexto das Orações; e

2) Avaliação de Significados e Extração de Informações.

A tabela 5.3.4 descreve a estrutura principal o algoritmo "Identificação de Comandos e Parâmetros".

Tabela 5.3.4 – Algoritmo "Identificação de Comandos e Parâmetros".

Passo	Descrição
01	Executar seção Capturar Erros de Comando Ausente.
02	Executar seção Avaliar Contexto da Oração.
03	Rótulo "Comando Mover", executar:
04	Avaliar Significado e Extrair Informações usando Caso Mover.
05	Retornar resultados como saída para o algoritmo.
05	Encerrar algoritmo.
06	Rótulo "Comando Rotacionar", executar:
07	Avaliar Significado e Extrair Informações usando Caso Rotacionar.
08	Retornar resultados como saída para o algoritmo.
09	Encerrar algoritmo.
10	Rótulo "Comando Rotear", executar:
11	Avaliar Significado e Extrair Informações usando Caso Rotear.
12	Retornar resultados como saída para o algoritmo.
13	Encerrar algoritmo.
14	Rótulo "Comando Desconhecido", executar:
15	Atribuir expressão "SEM COMANDO" à coluna de comando.
16	Atribuir mensagem de erro de comando ausente à coluna de mensagens.
17	Retornar resultados como saída para o algoritmo.
18	Encerrar algoritmo.
19	Rótulo "Comando Ambíguo", executar:
20	Atribuir vazio à coluna de comando.
21	Atribuir mensagem de erro de comando não identificado à coluna de mensagens.
22	Retornar resultados como saída para o algoritmo.
23	Encerrar algoritmo.
24	Rótulo "Fim", executar:
25	Retornar resultados (colunas) como saída para o algoritmo.
26	Encerrar algoritmo.

Tabela 5.3.5 – Seção "Capturar Erros de Comando Ausente" do algoritmo.

Passo	Descrição
01	Se o tamanho da matriz de *tokens* ou de *Tags* for igual à zero, executar:
02	Atribuir vazio à coluna de comando.
03	Atribuir mensagem de erro à coluna de mensagens.
04	Ir para Fim.
05	Fim do Se
05	Se o tamanho da matriz de *tokens* for diferente da matriz de *Tags*, executar:
06	Atribuir vazio à coluna de comando.
07	Atribuir mensagem de erro à coluna de mensagens.
08	Ir para Fim.
09	Fim do Se
10	Se não houver nenhum verbo na oração, executar:
11	Ir para "Comando Desconhecido".
12	Fim do Se
13	Se oração possui interjeição e tiver menos que três palavras, executar:
14	Ir para "Comando Desconhecido".
15	Fim do Se

Por ser este o ponto de entrada do algoritmo, inicialmente é feita uma verificação para capturar erros simples e previsíveis de ausência de comando informado ou possíveis falhas de funcionamento do sistema. Assim, não será utilizado nenhum recurso complexo do sistema com base em informações insuficientes ou incorretas, melhorando a qualidade de seu funcionamento. A tabela 5.3.5 descreve o trecho "Captura Erros de Comando Ausente".

Tabela 5.3.6 – Seção "Avaliar Contexto da Oração" do algoritmo.

Passo	Descrição
01	Para cada palavra ou *token* da oração, executar:
02	Se a palavra ou *token* constar do Léxico, executar:
03	Extrair do Léxico a contagem associada ao Caso Mover.
04	Somar o valor extraído ao de comparação da oração (para o Caso Mover).
05	Extrair do Léxico a contagem associada ao Caso Rotacionar.
05	Somar o valor extraído ao de comparação da oração (para o Caso Rotacionar).
06	Extrair do Léxico a contagem associada ao Caso Rotear.
07	Somar o valor extraído ao de comparação da oração (para o Caso Rotear).
08	Incrementar a contagem de palavras contextualizadas em um.
09	Fim do Se
10	Fim do Para
11	Se a contagem de palavras for maior que zero, calcular:
12	Probabilidade no Caso Mover = contagem no caso / contagem de palavras.
13	Probabilidade no Caso Rotacionar = contagem no caso / contagem de palavras.
14	Probabilidade no Caso Rotear = contagem no caso / contagem de palavras.
15	Fim do Se
16	Se a probabilidade em todos os Casos for igual a zero, executar:
17	Ir para Comando Desconhecido.
18	Fim do Se
19	Se a probabilidade para o Caso Mover for maior que todos os outros Casos, executar:
20	Ir para Comando Mover.
21	Fim do Se
22	Se a probabilidade para o Caso Rotacionar for maior que todos os outros Casos, executar:
23	Ir para Comando Rotacionar.
24	Fim do Se
25	Se a probabilidade para o Caso Rotear for maior que todos os outros Casos, executar:
26	Ir para Comando Rotear.
27	Fim do Se
28	Se a probabilidade de dois ou mais Casos forem iguais, executar:
29	Ir para Comando Ambíguo.
30	Fim do Se
31	Atribuir vazio à coluna de comando.
32	Atribuir mensagem de erro comando ausente à coluna de mensagens.
33	Ir para Fim.

5.3.6.1 Avaliação do Contexto das Orações

O objetivo é determinar em qual contexto geral se situa a oração como um todo, ou seja, qual o Esquema de Caso mais provável associado, dentro dos que foram previamente planejados para o sistema frente às funcionalidades apresentadas pelo robô e os serviços possíveis decorrentes da sua abstração.

É oportuno lembrar que o usuário também poderá usar o comando Mover para os lados. Se isto for identificado no texto, será preciso adicionar um comando de Rotacionar para Direita ou Esquerda antes do comando Mover. Assim, para simplificação, os Esquemas de Caso Mover e Rotacionar&Mover podem ser tratados por um único algoritmo.

A tabela 5.3.6 descreve o trecho do algoritmo intitulado "Avaliação do Contexto das Orações".

5.3.6.2 Avaliação de Significados e Extração de Informação

O objetivo é preencher os papéis previstos buscando as informações presentes no texto informado pelo usuário, uma vez que já se determinou qual, dentre os planejados para o sistema, é o Esquema de Caso de referência.

Para realizar a extração, as palavras presentes no texto precisam ter se significado avaliado e, em decorrência deste significado, determinar se os papéis do Esquema de Caso poderão ser preenchidos, se deverão ser preenchidos com valores previamente padronizados e gerar as mensagens de aviso ou erro pertinentes.

Um Léxico (dicionário) deverá ser usado para permitir esta avaliação. Ele foi elaborado de forma a representar o significado das palavras em linguagem natural de maneira conveniente e previamente estruturada que permita computar esse significado no seu contexto de utilização.

Já que é possível expressar uma mesma estrutura profunda de significado através de várias construções sintáticas com palavras em linguagem natural, ou seja, variadas estruturas superficiais, o algoritmos precisa ser flexível o suficiente para ser capaz de tolerar essa flexibilidade e mapear a estrutura profunda. Isto pode ser realizado com a associação da teoria dos Esquemas de Casos ("*Case Frames*") e um Léxico planejado com cuidado.

Convém observar que cada Esquema de Caso previsto requer a construção de trechos de código de programa adequado, que será o responsável pela execução organizada e controlada do que ele significa, quer seja uma ação ou um serviço.

Tais trechos de código de programa deverão constar das seguintes partes componentes do sistema:

- Avaliação do Contexto da Oração;
- Avaliação de Significados e Extração de Informação; e
- Lógica e Controle do Robô.

Os seguintes Esquemas de Casos foram planejados para o sistema:

a) Caso do Comando Mover;

b) Caso do Comando Rotacionar; e

c) Caso do Comando Rotear.

A tabela 5.3.8 descreve trecho comum a todos os algoritmos relativos aos Esquemas de Casos dos itens “a”, “b” e “c” acima, que foi destacado por conveniência e otimização de espaço.

Tabela 5.3.8 – Seção “Capturar Erros de Comando Ausente” do algoritmo.

Passo	Descrição
01	Se o tamanho da matriz de *tokens* ou de *Tags* for igual à zero, executar:
02	Atribuir vazio à coluna de comando.
03	Atribuir mensagem de erro à coluna de mensagens.
04	Ir para Fim.
05	Fim do Se
05	Se o tamanho da matriz de *tokens* for diferente da matriz de *Tags*, executar:
06	Atribuir vazio à coluna de comando.
07	Atribuir mensagem de erro à coluna de mensagens.
08	Ir para Fim.
09	Fim do Se

a) Algoritmo para o Caso do Comando “Mover”

O Esquema do Caso “Mover” foi planejado prevendo a existência de três papéis que precisam ser obrigatoriamente preenchidos:

- Distância;
- Sentido; e
- Velocidade.

Portanto, se não for possível conhecer todas as possíveis palavras e construções sintáticas capazes de expressar esses três papéis, será preciso buscar a maior parte das que sejam mais comumente usadas. As palavras deverão ser adicionadas ao Léxico, aonde serão vinculadas a parâmetros que permitirão o reconhecimento de seu significado e valor. Estes parâmetros devem ser verificados comparativamente para cada Esquema de Caso previsto pelo sistema, para torná-lo capaz de resolver principalmente o frequente problema da ambiguidade de significado.

A tabela 5.3.9 descreve a parte "Avaliação do Significado e Extração de Informação" do algoritmo para o Caso Mover.

Tabela 5.3.9 – Algoritmo para o Caso do comando "Mover".

Passo	Descrição
01	Executar seção Capturar Erros de Comando Ausente.
02	Executar seção Encontrar Sentido.
03	Executar seção Encontrar Distância.
04	Executar seção Encontrar Velocidade.
05	Rótulo "Fim", executar:
06	Compor o comando Mover com seus três parâmetros.
07	Compor as mensagens de aviso e erro, conforme for.
08	Verificar se há Rotacionar adicional antes do Mover a ser incluído.
09	Retornar resultados (colunas) como saída para o algoritmo.
10	Encerrar algoritmo.

A seção "Encontrar Sentido" do comando "Mover" será descrita a seguir.

É oportuno lembrar que o usuário também poderá instruir o robô para Mover para os lados. Se isto for identificado no texto, será preciso adicionar um comando de Rotacionar para Direita ou Esquerda antes do comando Mover. Assim, para simplificação, aos Esquemas de Caso Mover e Rotacionar&Mover podem ser tratados por um único algoritmo.

Na situação de ausência da indicação do sentido do movimento, o sistema emitirá mensagem de erro juntamente com aviso da solução paliativa que será assumir o movimento padrão para frente.

A tabela 5.3.10 descreve a seção "Encontrar Sentido" do Caso Mover.

Tabela 5.3.10 – Seção “Encontrar Sentido” do comando “Mover”.

Passo	Descrição
01	Para cada palavra ou *token* da oração, executar:
02	Se for Verbo, executar:
03	Se estiver no Léxico e significado implicar conteúdo de Sentido, executar:
04	Considerar o conteúdo de Sentido (ângulo).
05	Incrementar contagem de rotação Direita/Esquerda, conforme for.
05	Incrementar contagem de seguir para Frente/Trás, conforme for.
06	Fim do Se
07	Fim do Se
08	Se for Substantivo, Adjetivo, Advérbio ou Preposição, executar:
09	Se estiver no Léxico e significado implicar conteúdo de Sentido, executar:
10	Considerar o conteúdo de Sentido (ângulo).
11	Incrementar contagem de rotação Direita/Esquerda, conforme for.
12	Incrementar contagem de seguir para Frente/Trás, conforme for.
13	Fim do Se
14	Fim do Se
15	Fim do Se
16	Se contagem de seguir para Frente/Trás for maior que zero, executar:
17	Se o ângulo for múltiplo inteiro de 360°, então Mover é para frente.
18	Senão, Mover é para trás.
19	Fim do Se
20	Se contagem de rotação para Direita/Esquerda for maior que zero, executar:
21	Inserir comando Rotacionar adicional.
22	Ir para Distância.
23	Fim do Se
24	Se ângulo for diferente de zero, executar:
25	Caso igual a +90° ou igual a -90°, executar:
26	Inserir comando Rotacionar adicional.
27	Caso igual a 180°, executar:
28	Mover é para trás.
29	Caso 360°, executar:
30	Mover é para frente
31	Ir para Distância.
32	Fim do Se
33	Assumir sentido padrão (para frente) para o comando Mover.
34	Incluir mensagem de erro de sentido ausente e aviso da solução paliativa.

Na seção “Encontrar Distância” do comando “Mover” descrita na tabela 5.3.11, foi suprimida a indicação do rótulo, apenas para facilitar a visualização.

Na situação de ausência da unidade de medida para a distância, o sistema emitirá mensagem de erro juntamente com aviso da solução paliativa que será assumir as unidades de medida a seguir:

- Unidade padrão do sistema (um quadrado, conforme o ajuste da escala da planta) para valores menores que dez;

- Centímetro para valores entre dez e cem; e
- Milímetro para valores maiores que cem.

Tabela 5.3.11 – Seção "Encontrar Distância" do comando "Mover".

Passo	Descrição
01	Para cada palavra ou *token* da oração, executar:
02	Se for Número, executar:
03	Considerar valor numérico para distância a percorrer.
04	Sair do Para
05	Fim do Se
05	Fim do Para
06	Se distância a percorrer for igual á zero, executar:
07	Assumir valor padrão para distância a percorrer.
08	Incluir mensagem de erro de distância ausente à coluna de mensagens.
09	Ir para Velocidade.
10	Fim do Se
11	Para cada palavra ou *token* da oração, executar:
12	Se não for Substantivo, Preposição ou Interjeição, pular para a próxima.
13	Se estiver no Léxico e o significado implicar conteúdo de Distância, executar:
14	Considerar o conteúdo de Distância como unidade de medida.
15	Incrementar a contagem de palavras encontradas.
16	Fim do Se
17	Fim do Se
18	Se contagem de palavras encontradas for maior que zero, executar:
19	Multiplicar a distância a percorrer pelo conteúdo da unidade de medida.
20	Atribuir o resultado à distância a percorrer.
21	Senão
22	Caso a distância a percorrer menor que 10, executar:
23	Considerar a unidade de medida como sendo a distância padrão.
24	Calcular por multiplicação e atribuir à distância a percorrer.
25	Caso a distância a percorrer esteja entre 10 e 100, executar:
26	Considerar a unidade de medida como sendo "cm".
27	Calcular por multiplicação e atribuir à distância a percorrer.
28	Caso a distância a percorrer seja maior que 100, executar:
29	Considerar a unidade de medida como sendo "mm".
30	Calcular por multiplicação e atribuir à distância a percorrer.
31	Incluir mensagens de erro e de aviso sobre a solução paliativa.
32	Fim do Se

Na seção "Encontrar Velocidade" do comando "Mover" descrita a seguir, foi suprimida a indicação do rótulo, apenas para facilitar a visualização.

A velocidade esperada deverá ser informada de forma verbal pelo usuário, e não numérica, através do significado implícito das palavras como verbos, adjetivos e advérbios, por exemplo. Não serão reconhecidas unidades de medida como km/h, m/s outros, uma vez que foi pressuposto que o usuário perderia a noção clara dessas

unidades e velocidades de forma que pudessem ter aplicação útil. O escorregamento das rodas do robô sobre o solo também não poderiam garantir a precisão destes valores, assim como a condição de carga da bateria e outras imprecisões do projeto. Então, somente serão reconhecidas indicações subjetivas de velocidade generalizadas simplesmente em três níveis: Normal, Devagar e Rápido.

Se porventura o usuário utilizar expressões confusas como “corra devagar”, “fugir lentamente” ou “bem lentamente avance rápido”, o sistema calculará uma média dos valores implícitos em cada palavra, tentando satisfazer a curiosidade do usuário.

A tabela 5.3.12 descreve a seção “Encontrar Velocidade” do Caso Mover.

Tabela 5.3.12 – Seção “Encontrar Velocidade” do comando “Mover”.

Passo	Descrição
01	Para cada palavra ou *token* da oração, executar:
02	Se for Verbo, estiver no Léxico e significado implicar conteúdo de Velocidade, executar:
03	Considerar conteúdo de Velocidade encontrado.
04	Incrementar a contagem de palavras encontradas.
05	Fim do Se
05	Se for Adjetivo, Advérbio, Prefixo ou Interjeição e estiver no Léxico, executar:
06	Se significado implicar conteúdo de Velocidade, executar:
07	Considerar conteúdo de Velocidade encontrado, fazendo média dos valores.
08	Incrementar a contagem de palavras encontradas.
09	Fim do Se
10	Fim do Se
11	Fim do Para
12	Se a contagem de palavras encontradas for maior que zero, executar:
13	Se a Velocidade considerada for igual a zero, executar:
14	Assumir valor padrão para a Velocidade e ir para Fim.
15	Senão
16	Assumir a Velocidade considerada e ir para Fim.
17	Fim do Se
18	Fim do Se
19	Assumir valor padrão para a Velocidade.

b) Algoritmo para o Caso do comando “Rotacionar”

O Esquema do Caso “Rotacionar” foi planejado prevendo a existência de apenas um papel que precisam ser obrigatoriamente preenchidos:

- Sentido.

Do ponto de vista do robô Curumim, esta ação ou comando possui um parâmetro adicional de Velocidade para a rotação. Contudo, alguns experimentos

mostraram que velocidades diferentes provocam erros diferentes na rotação em decorrência da inércia, tanto no início da ação quando ao final.

A tabela 5.3.13 descreve a parte "Avaliação do Significado e Extração de Informação" do algoritmo para o Caso Rotacionar.

Tabela 5.3.13 – Algoritmo para o Caso do comando "Rotacionar".

Passo	Descrição
01	Executar seção Capturar Erros de Comando Ausente.
02	Executar seção Encontrar Sentido.
03	Rótulo "Fim", executar:
04	Retornar resultados (colunas) como saída para o algoritmo.
05	Encerrar algoritmo.

A seção "Encontrar Sentido" do comando "Rotacionar" será descrita a seguir.

Na situação de ausência da indicação do sentido do movimento, o sistema emitirá mensagem de erro juntamente com aviso da solução paliativa que será assumir rotação padrão para a direita ou +90°.

Tabela 5.3.14 – Seção "Encontrar Sentido" do comando "Rotacionar".

Passo	Descrição
01	Para cada palavra ou *token* da oração, executar:
02	Se não for Substantivo, Adjetivo, Advérbio ou Preposição, pular para a próxima;
03	Se estiver no Léxico e significado implicar conteúdo de Sentido, executar:
04	Considerar o conteúdo de Sentido (ângulo).
05	Fim do Se
05	Fim do Para
06	Se foi encontrado conteúdo de Sentido, executar:
07	Compor o comando Rotacionar com o conteúdo encontrado.
08	Inserir mensagens de erro e aviso da solução paliativa.
09	Senão
10	Compor o comando Rotacionar com valor padrão (direita ou +90°).
11	Fim do Se

Se porventura o usuário utilizar mais de uma palavra indicando sentido para rotacionar, o sistema considerará somente a última que aparecer na oração, desprezando as demais. Isto se dá em decorrência da lógica de funcionamento do algoritmo.

A tabela 5.3.14 descreve a seção "Encontrar Sentido" do Caso Rotacionar.

c) Algoritmo para o Caso do comando “Rotear”

O Esquema do Caso “Rotear” foi planejado prevendo a existência de apenas um papel que precisam ser obrigatoriamente preenchidos:

- Local (destino); e
- Velocidade.

Este se refere a um Esquema de Caso associado a um serviço, diferentemente de “Mover” e “Rotacionar” que se referem a funções simples do robô. O comando “Rotear” tentará buscar o melhor caminho para atingir um Local (destino) indicado pelo usuário e, caso seja encontrado, reverterá uma sequência de comandos “Mover” e “Rotacionar” apropriada para percorrer este caminho.

Portanto, o comando “Rotear” emprega vários algoritmos distintos de forma coordenada para produzir o resultado desejado.

A tabela 5.3.15 descreve a parte “Avaliação do Significado e Extração de Informação” do algoritmo para o Caso Rotear.

Tabela 5.3.15 – Algoritmo para o Caso do comando “Rotear”.

Passo	Descrição
01	Executar seção Capturar Erros de Comando Ausente.
02	Executar seção Encontrar Velocidade e Local (destino).
03	Se for encontrado Local (destino), executar:
04	Executar seção Colocar Alvo na Planta.
05	Senão
06	Incluir mensagem de erro de falta de Local.
07	Ir para Fim.
08	Fim do Se
09	Executar algoritmo Buscar Caminho.
10	Executar função Recalcular Planta.
11	Executar algoritmo Rumar Para (resulta em sequência de comandos Mover e Rotacionar).
12	Rótulo “Fim”, executar:
13	Retornar resultados (colunas) como saída para o algoritmo.
14	Encerrar algoritmo.

A seção “Encontrar Velocidade e Local” do comando “Rotear” será descrita a seguir. O Local refere-se ao destino a ser atingido e poderá ser mencionado também como Local, Destino ou Alvo ao longo este texto.

Na situação de ausência da indicação de velocidade, o sistema emitirá mensagem de erro juntamente com aviso da solução paliativa que será assumir velocidade padrão normal.

Se porventura o usuário utilizar mais de uma palavra indicando velocidade, o sistema considerará somente a última que aparecer na oração, desprezando as demais. Isto se dá em decorrência da lógica de funcionamento do algoritmo e optou-se por manter assim, diferentemente da solução adotada para o comando “Mover”.

Com relação ao Local, este significa a coordenada de um ponto na Planta do ambiente. Foi implementada a extração de coordenadas somente a partir do Léxico, ou seja, foram dados nomes a alguns pontos da planta.

Desde que as funcionalidades de Busca de Rota do sistema robótico proposto apresentam certa independência das funcionalidades de Linguagem Natural, uma sugestão de desenvolvimento futuro é o estudo para permitir nomear os pontos diretamente na Planta bem como os meios de extrair estes nomes de forma que se tornem disponíveis aos algoritmos de Processamento da Linguagem Natural.

A tabela 5.3.16 descreve a seção “Encontrar Velocidade e Local” do Caso Rotear.

Tabela 5.3.16 – Seção “Encontrar Velocidade e Local” do comando “Rotear”.

Passo	Descrição
01	Para cada palavra ou *token* da oração, executar:
02	Se for Verbo, estiver no Léxico e significado implicar conteúdo de Sentido, executar:
03	Considerar o conteúdo de Velocidade.
04	Incrementar a contagem de palavras encontradas.
05	Fim do Se
05	Se for Substantivo, Adjetivo, Advérbio ou Preposição e estiver no Léxico, executar:
06	Se significado implicar conteúdo de Local, executar:
07	Considerar conteúdo de Local (coordenadas) para o Alvo.
08	Fim do Se
09	Se significado implicar conteúdo de Velocidade, executar:
10	Considerar o conteúdo de Velocidade.
11	Fim do Se
12	Fim do Se
13	Fim do Para

A seção “Colocar Alvo na Planta” do comando “Rotear” será descrita a seguir.

Na situação de ausência da indicação de Local, o sistema emitirá mensagem de erro. Nenhuma ação será tomada uma vez que é impossível assumir um valor padrão para destino de uma rota.

Desde que sejam encontradas coordenadas no Léxico, é feita verificação para garantir que estejam dentro da Planta. Se este erro for encontrado, será emitida mensagem e o algoritmo será abortado.

Se as coordenadas coincidirem com a posição atual do robô, será emitida mensagem de erro, sendo que o algoritmo também será abortado.

A tabela 5.3.17 descreve a seção “Colocar Alvo” na Planta do Caso Rotear.

Tabela 5.3.17 – Seção “Colocar Alvo na Planta” do comando “Rotear”.

Passo	Descrição
01	Se coordenada X do alvo estiver fora da planta, executar:
02	Incluir mensagem de erro de alvo fora da planta.
03	Ir para Fim.
04	Fim do Se
05	Se coordenada Y do alvo estiver fora da planta, executar:
05	Incluir mensagem de erro de alvo fora da planta.
06	Ir para Fim.
07	Fim do Se
08	Se coordenadas do alvo coincidirem com a do robô, executar:
09	Incluir mensagem de aviso de que robô já está sobre o alvo.
10	Ir para Fim.
11	Fim do Se
12	Preparar Planta para executar busca.
13	Remover da Planta alvo anterior.
14	Definir na Planta o novo alvo.

O funcionamento do algoritmo Buscar Caminho é detalhado na seção 2.8 (Algoritmo de busca do melhor caminho), sendo que foi implementado no sistema robótico somente o A* (A-Estrela).

5.3.7 ALGORITMO “RUMAR PARA”

Para que o caminho encontrado pelo algoritmo A* (A-Estrela) seja percorrido pelo robô é necessário antes convertê-lo em uma sequência apropriada de comandos Mover e Rotacionar compatíveis com o robô. Assim, os obstáculos do ambiente poderão ser contornados conforme previamente determinado.

A implementação no sistema robótico foi de acordo com o Curumim. Outros robôs poderão dispor de recursos de deslocamento diferentes, como movimentação omnidirecional (em todas as direções), por exemplo. Neste caso, uma solução diferente

de conversão do caminho encontrado para comandos de movimentação pode ser elaborada, conforme for o caso.

Sua entrada é uma variável de número inteiro que conterá a velocidade que será usada nos comandos Mover. Porém, para funcionamento, o algoritmo retirará as demais informações para funcionar diretamente da Planta do ambiente (existência ou não do Alvo, a existência ou não de um caminho até ele bem como o caminho em si).

Sua saída é uma matriz unidimensional de *strings* (cadeia de caracteres) contendo os comandos Mover e Rotacionar, já com seus respectivos parâmetros de configuração, organizados na sequência adequada para fazer o robô percorrer o caminho previsto na anteriormente pelo *software*.

A tabela 5.3.18 descreve a estrutura principal o algoritmo “Rumar Para”.

Tabela 5.3.18 – Algoritmo “Rumar Para”.

Passo	Descrição
01	Executar seção “Avaliar condições e preparar”.
02	Executar seção “Transformar caminho em comandos”.
03	Retornar resultado como saída para o algoritmo.
04	Encerrar o algoritmo.

A seção “Avaliar condições e preparar” do algoritmo “Rumar para” será descrita a seguir.

Como avaliação das condições, será verificado se existe um Alvo e existe um caminho possível até ele. Se o número de passos até o alvo for maior que o número de quadrados da planta, então não existe um caminho possível até o alvo.

Esta verificação decorre do funcionamento do algoritmo de busca de caminho. Após sua execução com sucesso, os pontos do caminho, que vão do Robô até o Alvo, são organizados em ordem crescente de passos, sendo que o último passo atinge o Alvo.

No ciclo de execução, o algoritmo atribui inicialmente uma posição ordinal para o Alvo muito além da quantidade máxima de pontos da planta e, após a determinação do caminho, ajusta esta posição ordinal para o valor correto, quando um caminho é encontrado. Ou então, aborta a execução após uma quantidade finita de ciclos de tentativas para determinar o caminho, se nenhum caminho for encontrado, sem alterar a posição ordinal do Alvo.

Se houver de fato um caminho ato o Alvo, é criada uma lista interna para controlar o funcionamento do algoritmo, com igual número de passos ao do caminho encontrado pelo *software* anteriormente. Em cada elemento dessa lista são colocadas as coordenadas dos pontos respectivamente a cada passo do caminho e na ordem exata em que devem ser percorridos.

Essas informações são extraídas da própria Planta, diretamente a partir da memória do sistema robótico.

A tabela 5.3.19 descreve a seção “Avaliar condições e preparar” do algoritmo “Rumar Para”.

Tabela 5.3.19 – Seção “Avaliar condições e preparar” do algoritmo “Rumar Para”.

Passo	Descrição
01	Para cada ponto da planta, executar:
02	Se identificação do ponto for igual a Alvo, executar:
03	Se número de passos for maior que Qte. De Linhas X Qte. De Colunas, executar:
04	Executar Recalcular Planta.
05	Se número de passos for maior que Qte. De Linhas X Qte. De Colunas, executar:
05	Sair do Para.
06	Fim do Se
07	Fim do Se
08	Atribuir número de passos à capacidade da Lista do Caminho.
09	Iniciar os elementos da Lista do Caminho com zero.
10	Colocar coordenadas do Alvo na última posição da Lista de Caminho.
11	Sair do Para.
12	Fim do Se
13	Fim do Para
14	Se número de passos for maior que Qte. De Linhas X Qte. De Colunas, executar:
15	Atribuir aviso de “caminho inexiste" ao resultado.
16	Retornar resultado como saída para o algoritmo.
17	Encerrar o algoritmo.
18	Fim do Se
19	Para cada ponto da planta, executar:
20	Se identificação do ponto for igual a Caminho, executar:
21	Colocar coordenadas dos pontos Lista do Caminho, na mesma ordem dos passos.
22	Fim do Se
23	Fim do Para

A seção “Transformar caminho em comando” do algoritmo “Rumar para” será descrita a seguir.

A transformação do caminho descrito por uma sequência de pontos em ordem em uma sequência de comandos Mover e Rotacionar é feita comparando as coordenadas de dois pontos consequentes.

Se a diferença entre suas coordenadas X e a diferença entre suas coordenadas Y não significar mudança do ângulo de orientação do robô em relação à planta, então o movimento é em linha reta e a distância entre os pontos será somada à distância total a ser percorrida.

Se a diferença entre suas coordenadas X e a diferença entre suas coordenadas Y significar mudança do ângulo de orientação do robô em relação à planta, então será preciso enviar comando de Rotacionar ao robô, no ângulo equivalente a diferença necessária para atingir a orientação desejada. Antes, porém, será preciso enviar um comando Mover para percorrer uma distância em linha reta. Isto decorre do fato de que uma rotação define o fim de uma reta anterior e o início de uma reta seguinte.

Tabela 5.3.20 – Seção "Transformar caminho em comando" do algoritmo "Rumar Para".

Passo	Descrição
01	Para cada ponto da Lista do Caminho, executar:
02	Se for o primeiro ponto, executar:
03	Atribuir Posição Atual do robô à variável Última Posição
04	Atribuir Orientação Atual do robô à variável de Último Ângulo.
05	Senão
05	Atribuir coordenada do ponto à variável Posição Nova.
06	Determinar o Ângulo Novo comparando coordenadas da Última Posição e a Nova.
07	Determinar distância entre Última Posição e a Posição Nova.
08	Se o Último Ângulo e o Ângulo Novo forem diferentes, executar:
09	Se Soma da Distância em Reta for maior que zero, executar:
10	Inserir comando Mover nesta Soma da Distância em Reta ao resultado.
11	Fim do Se
12	Zerar a Soma da Distância em Reta.
13	Calcular a diferença entre Ângulo Novo e Último Ângulo.
14	Inserir comando Rotacionar nesta Diferença de Ângulo calculada ao resultado.
15	Fim do Se
16	Atribuir coordenada do ponto à variável Última Posição.
17	Adicionar a distância entre a Última Posição e a Nova à Soma da Distância em Reta.
18	Se for o último ponto, executar:
19	Inserir comando Mover na Soma da Distância em Reta ao resultado.
20	Fim do Se
21	Fim do Se
22	Fim do Para

É interessante observar que um comando Mover é gerado e enviado ao robô somente quando um comando Rotacionar é identificado ou quando o Alvo é encontrado no final do caminho. Um comando Rotacionar é geralmente necessário como primeiro comando (sem existir um comando Mover anterior) para colocar o robô na orientação correta em relação à Planta de acordo com o caminho a ser percorrido.

A tabela 5.3.20 descreve a seção “Transformar caminho em comando” do algoritmo “Rumar Para”.

5.3.8 ALGORITMO “ADIÇÃO À LISTA DE COMANDOS”

Uma vez dispondo da sequência de comandos identificados à partir da instrução fornecida pelo usuário em linguagem natural, esses comandos precisam ser colocados em uma lista, juntamente com seus parâmetros de configuração e suas mensagens de aviso ou erro para serem submetidas à avaliação do usuário. Assim, ele poderá confirmar se estão corretos e executá-los ou então providenciar as correções necessárias, no texto em linguagem natural, para obter a interpretação, e, por conseguinte, os resultados desejados.

Sua entrada é uma matriz bidimensional de *strings* (cadeia de caracteres). A primeira coluna contém os comandos identificados, já com seus respectivos parâmetros de configuração, organizados na mesma sequência em que aparecem no texto fornecido pelo usuário. A segunda coluna contém as mensagens de erro e avisos identificadas pelo sistema durante o processamento, respectivamente a cada comando identificado.

Sua saída é alteração do conteúdo do elemento constante na Interface em Linguagem Natural, responsável pela exibição da sequência de comandos interpretados, que também será usado também durante a execução propriamente dita.

A tabela 5.3.21 descreve o algoritmo “Adição à Lista de Comandos”.

Tabela 5.3.21 – Algoritmo “Adição à Lista de Comandos”.

Passo	Descrição
01	Para cada Comando e Parâmetro identificado, executar:
02	Adicionar uma nova linha na Lista de Comandos.
03	Adicionar na primeira coluna o Comando com seu Parâmetro.
04	Se houver, adicionar na segunda coluna a Mensagem de Aviso ou Erro.
05	Fim do Para

5.3.9 PROCEDIMENTO “EXECUTAR COMANDO”

Dispondo da lista correta com os comandos que foram interpretados do texto em linguagem natural, o usuário poderá escolher entre algumas formas de executar a

sequência. São recursos da Interface em Linguagem Natural serão mostrados nos itens seguintes.

Eles se utilizam procedimentos e funções de *software* recursivamente para poderem apresentar os resultados esperados. Todos fazem parte do bloco de *Interface com o Robô (IFR)* e desempenham a *Lógica e Controle do Robô*.

O procedimento "Executar Comando" é responsável por separar um determinado comando da lista e encaminhá-lo para o robô, através do procedimento apropriado que cuidará disso (procedimento "Enviar Comando"), ao lado que apresentará algumas informações de acompanhamento ao usuário, na janela.

Sua entrada é uma variável do tipo número inteiro, que indica qual o comando da sequência que deverá ser executado.

Sua saída não é propriamente estruturada na forma de variável, mas sim as ações que serão executadas: o comando do robô em sim e a exibição de uma mensagem de acompanhamento ao usuário.

O tempo de espera de 300 milissegundos adicionado é mera conveniência para proporcionar maior conforto visual ao usuário no caso de execução simulada, sem que o robô esteja realmente conectado e recebendo os comandos.

A tabela 5.3.22 descreve o procedimento "Executar Comando".

Tabela 5.3.22 – Procedimento "Executar Comando".

Passo	Descrição
01	Se Comando Indicado estiver além da Lista de Comandos, executar:
02	Exibir mensagem ao usuário: "Erro: comando não existe na sequência".
03	Sair do procedimento.
04	Fim do Se
05	Pegar comando e seu parâmetro da lista e separa o comando da mensagem aviso/erro.
06	Exibir mensagem ao usuário: "Executando: " seguido do comando após os dois pontos.
07	Compor os parâmetros do procedimento "Enviar Comando" e o disparar.
08	Aguardar 300 milissegundos.
09	Sair do procedimento.

5.3.10 PROCEDIMENTO "ENVIAR COMANDO"

Para que seja possível executar um comando, algumas ações devem ser executadas de forma coordenada para que os resultados esperados sejam obtidos:

- Ler sensores de obstáculo, no caso de comando de movimentação;
- Verificar se distância até obstáculo permite movimento;
- Executar o comando de movimento ou rotação;
- Atualizar os indicadores e registradores internos de posição e ângulo do robô; e
- Controlar se execução deve ser abortada em decorrência de obstáculo encontrado de imprevisto.

A entrada deste procedimento é composta de: uma variável do tipo *string* (cadeia de caracteres) e uma variável do tipo número inteiro. A variável string contém o comando previamente separado da lista e a variável numérica refere-se ao respectivo comando da lista de comandos.

Sua saída não é propriamente estruturada na forma de variável, mas sim as ações que serão executadas.

Observar que a distância máxima segura para movimento é aquela menor que o limite mínimo do sensor, caso haja indicação de um limite máximo diferente de infinito (no caso, 9999 no Sistema Robótico, ou seja, maior que a maior distância detectável pelos sensores do robô). Um limite máximo menor que o considerado infinito significará que foi encontrado um obstáculo entre os limites mínimo e máximo e um movimento entre estes limites seria arriscado.

Por serem funcionalidades secundárias, relacionadas meramente a recursos visuais e, portanto, além do escopo do conteúdo desta dissertação, não serão detalhados os procedimentos de exibição na janela (indicadores de posição e ângulo do robô e posição do robô na planta).

A tabela 5.3.23 descreve o procedimento "Enviar Comando".

Tabela 5.3.23 – Procedimento “Enviar Comando”.

Passo	Descrição
01	Executa separação do Comando de seus Parâmetros.
02	Se Comando for igual a “MOVE”, executar:
03	Executa separação da Distância e da Velocidade dentre os Parâmetros.
04	Executa formatação numérica apropriada da Distância e da Velocidade.
05	Se Robô Conectado e Leitura de Sensores habilitada, executar:
06	Ler sensores.
07	Fim do Se
08	Se Distância for maior que zero, executar:
09	Sentido do Movimento: para frente.
10	Atribui limite mínimo de distância conforme sensor dianteiro.
11	Atribui limite máximo de distância conforme sensor dianteiro.
12	Senão, executar:
13	Sentido do Movimento: para trás.
14	Atribui limite mínimo de distância conforme a leitura máxima dos sensores traseiros.
15	Atribui limite máximo de distância conforme a leitura mínima dos sensores traseiros.
16	Fim do Se
17	Se Limite Mínimo for maior que Distância, executar:
18	Se Robô Conectado, executar:
19	Executa comando “AndarReto” com Distância e Velocidade.
20	Fim do Se
21	Atualiza indicadores de ângulo e posição do robô na interface.
22	Atualiza posição do robô na planta.
23	Fim do Se
24	Se Limite Mínimo for menor que Distância e Limite Máximo for infinito, executar:
25	Se Robô Conectado, executar:
26	Executa comando “AndarReto” com Distância e Velocidade.
27	Fim do Se
28	Atualiza indicadores de ângulo e posição do robô na interface.
29	Atualiza posição do robô na planta.
30	Fim do Se
31	Se Limite Mínimo for menor que Distância e Limite Máximo não for infinito, executar:
32	Exibe mensagem de erro sobre existência de obstáculo e o movimento.
33	Marca Situação da Execução como abortada.
34	Fim do Se
35	Fim do Se
36	Se Comando for igual a “ROTATE”, executar:
37	Executa separação do Sentido dentre os Parâmetros.
38	Executa formatação numérica apropriada do Sentido.
39	Se Sentido for menor ou igual a 90 graus, executar:
40	Se Robô Conectado, executar:
41	Executa comando “Rotacionar” com Sentido e velocidade padronizada.
42	Fim do Se
43	Atualiza indicadores de ângulo e posição do robô na interface.
44	Atualiza posição do robô na planta.
45	Senão
46	Se Robô Conectado, executar:
47	Executa comando “Rotacionar” com 90 graus e velocidade padronizada.
48	Executa comando “Rotacionar” com Sentido-90 graus e velocidade padronizada.
49	Fim do Se
50	Atualiza indicadores de ângulo e posição do robô na interface.
51	Atualiza posição do robô na planta.
52	Fim do Se
53	Fim do Se
54	Sair do procedimento

5.3.11 PROCEDIMENTO "EXECUTAR TUDO"

Este procedimento dispara a execução de todos os comandos da lista em sequencia, um após o outro. Ao lado disso, verifica se a execução deve ser abortada em decorrência de algum imprevisto e apresenta as mensagens de acompanhamento apropriadas ao usuário.

Não há uma entrada estruturada para este procedimento. Trata-se de um evento disparado pelo usuário através do botão "Executar" na Interface em Linguagem Natural.

Igualmente inexiste uma saída estruturada senão os eventos que resultarão.

A tabela 5.3.24 descreve o procedimento "Executar Tudo".

Tabela 5.3.24 – Procedimento "Executar Tudo".

Passo	Descrição
01	Para cada Comando da lista de comandos, executar:
02	Se Situação da Execução for Não Abortada, executar:
03	Executa procedimento "Executar Comando" para comando atual.
04	Fim do Se
05	Fim do Para cada.
06	Se Situação da Execução for Não Abortada, executar:
07	Exibir mensagem ao usuário de execução terminada.
08	Senão
09	Exibe mensagem de erro sobre execução abortada antes do término.
10	Fim do Se
11	Refresca exibição da figura do robô na janela.
12	Sair do procedimento.

5.3.12 PROCEDIMENTO "EXECUTAR PARCIAL"

Este procedimento dispara a execução os comandos da lista em sequência, um após o outro, mas somente a partir do ponto atual e não do começo da lista. Ao lado disso, verifica se a execução deve ser abortada em decorrência de algum imprevisto e apresenta as mensagens de acompanhamento apropriadas ao usuário.

Não há uma entrada estruturada para este procedimento. Trata-se de um evento disparado pelo usuário através do botão "Executar Parcial" na Interface em Linguagem Natural.

Igualmente inexiste uma saída estruturada senão os eventos que resultarão.

A tabela 5.3.25 descreve o procedimento “Executar Parcial”.

Tabela 5.3.25 – Procedimento “Executar Parcial”.

Passo	Descrição
01	Para cada Comando da lista de comandos a partir do atual, executar:
02	Se Situação da Execução for Não Abortada, executar:
03	Executa procedimento “Executar Comando” para comando atual.
04	Fim do Se
05	Fim do Para cada.
06	Se Situação da Execução for Não Abortada, executar:
07	Exibir mensagem ao usuário de execução terminada.
08	Senão
09	Exibe mensagem de erro sobre execução abortada antes do término.
10	Fim do Se
11	Refresca exibição da figura do robô na janela.
12	Sair do procedimento.

5.3.13 PROCEDIMENTO “EXECUTAR PASSO”

Este procedimento dispara a execução de um único comando presente na lista, ou seja, o que está no ponto atual. Ao lado disso, apresenta as mensagens de acompanhamento apropriadas ao usuário e atualização.

Não há uma entrada estruturada para este procedimento. Trata-se de um evento disparado pelo usuário através do botão “Executar Passo” na Interface em Linguagem Natural.

Igualmente inexiste uma saída estruturada senão os eventos que resultarão.

A tabela 5.3.26 descreve o procedimento “Executar Passo”.

Tabela 5.3.26 – Procedimento “Executar Passo”.

Passo	Descrição
01	Se existir algum comando selecionado na lista, executar:
02	Executa procedimento “Executar Comando” para comando atual.
03	Se ordinal do comando atual for menor que último, executar:
04	Selecionar comando seguinte.
05	Fim do Se
06	Senão
07	Executa procedimento “Executar Comando” para primeiro comando da lista.
08	Selecionar comando seguinte.
09	Fim do Se
10	Exibir mensagem ao usuário de execução terminada.
11	Atualiza posição do robô na planta.
12	Sair do procedimento.

5.4 A ELABORAÇÃO DA INTERFACE

Por ser a principal via de interação do usuário com o sistema robótico, para sua concepção buscou-se uma forma direta e simples de apresentar e solicitar as informações necessárias para operação e compreensão do Sistema Robótico proposto.

Como forma de simplificar, tanto do ponto de vista de programação, quanto do ponto de vista do aprendizado e operação por parte do usuário, optou-se por uma solução de janela única, ou seja, todas as informações têm entrada e saída na mesma janela principal.

Com relação ao fluxo de informações entre o usuário e o sistema, a necessidade primordial é que o sistema seja capaz de receber as instruções na forma de linguagem natural textual do usuário, apresente o resultado da interpretação dessas instruções e ofereça a oportunidade do usuário corrigir o que for necessário ou então disparar a execução.

Para receber as instruções do usuário, é preciso uma caixa de texto e para apresentar o resultado da interpretação é preciso uma área para listar os comandos e seus respectivos parâmetros de configuração.

Uma vez que a interface foi elaborada no estilo de um formulário tipo *Windows Forms*, alguns elementos acessórios são necessários para controla o seu funcionamento. Igualmente, é necessário dispor de maneiras de controlar outras funções do sistema robótico como a comunicação com o robô.

Assim, foram acrescentados botões de controle para:

- Indicar o fim da entrada das instruções em linguagem natural e disparar o processamento para interpretação do texto;
- Controlar a execução da sequência (lista) de comandos interpretados; e
- Habilitar, desabilitar e testar a conexão de comunicação com o robô.

Outra funcionalidade prevista para o sistema, como forma de agregar mais utilidade e opções de serviço, é a capacidade de manter um mapa ou planta do local aonde o robô se movimentará. Portanto, uma área específica foi reservada para apresentação e alteração do conteúdo de memória responsável pelo mapeamento, bem como se fizeram necessários botões para ser possível guardar e recuperar plantas bem

como refrescar a exibição dos dados da memória na respectiva área reservada dentro da janela/formulário.

Como forma de exibir ao usuário a localização e a orientação mais precisas do robô, e também permitir que este pudesse interagir com o conteúdo da memória do sistema responsável pelo controle destes parâmetros, acrescentou-se campos numéricos capazes de exibição como edição. Como decorrência, e também por conveniência do usuário, foi acrescentado um botão para fazer esses parâmetros retornarem ao estado inicial ou zerado.

Um último recurso previsível, que poderia tornar as condições de experimentação mais flexíveis, seria acrescentar campos numéricos para exibir e editar o conteúdo da memória responsável pelo registro da leitura dos sensores de obstáculo do robô. Também por decorrência e conveniência, foram acrescentados botões para disparo manual da leitura dos sensores, retorno geral dos valores para a condição inicial e habilitar ou não o uso dos sensores durante a execução da sequência de comandos.

Por fim, restou a questão da organização desses diversos elementos. O ponto de partida inicial foi o sentido natural de leitura de textos e formulários em língua portuguesa, ou seja, de cima para baixo e da esquerda para a direita. Neste caso, a entrada das instruções em linguagem natural deveria ocorrer na parte superior do formulário e a resposta do sistema (resultado da interpretação) seria apresentada abaixo. Os respectivos botões de controle foram colocados ou à esquerda desses elementos ou abaixo.

Como o tamanho da tela da maioria dos computadores menor na altura e maior na largura, surgiu um problema de limitação de espaço para a diagramação dos demais elementos previstos. O conceito proposto para a interface é de que a janela de comunicação deveria ser única, simples e direta, neste caso, isto significa que a disposição conveniente de todas as informações deveria ser com tudo visível na tela do computador, sem que houvesse a necessidade de uso da barra de rolagem. Assim, com um simples movimento dos olhos o usuário poderia visualizar o conteúdo desejado.

A solução foi agrupar o maior número possível de campos de informação no sentido horizontal na janela. Assim, a área para a planta do local foi disposta ao lado direito da caixa de listagem da sequência de comandos e, logo abaixo, da esquerda para a direita, foram distribuídos os campos numéricos e botões de controle da leitura dos

sensores, da localização e orientação do robô e os botões das funcionalidades acessórias da planta.

5.4.1 OBSERVAÇÕES SOBRE O DESENVOLVIMENTO

A maneira como ocorreu a concepção e o desenvolvimento da interface transcorreu temporalmente como descrito anteriormente nesta seção. A interface foi desenvolvida em ondas sucessivas e à medida que a necessidade de acesso aos parâmetros de controle foi surgindo durante o desenvolvimento do *software*.

Algumas versões mais simples foram propostas inicialmente para teste e avaliação de utilidade. Primeiramente, em torno da necessidade primordial de entrada das instruções em linguagem natural e listagem do resultado da sua interpretação. Com o acréscimo da funcionalidade de rota inteligente até um destino nomeado, a área para a planta precisou ser acrescentada.

Durante os testes e desenvolvimento da capacidade de rota inteligente surgiu a necessidade de permitir que o sistema trabalhasse sem que o robô estivesse conectado. Com isto, foi possível agregar a possibilidade de simulação virtual do funcionamento. E foram acrescentados os botões e desenvolvidos os respectivos procedimentos de *software* para controle da conexão com o robô.

Logo depois, surgiu a necessidade para exibir e controlar a posição de orientação do robô e, em seguida, para exibir e controlar os sensores de obstáculos. Neste caso, optou-se por manter as informações de localização e orientação do robô o mais próximo possível da planta do local, já que esses dados estão dentro do mesmo âmbito de funcionalidade (um completa o outro para perfazer a informação sobre onde no ambiente está o robô e onde estará com a execução do próximo passo).

5.5 A ELABORAÇÃO DO *SOFTWARE*

Por ser o *Software* de Inteligência e Controle um conjunto de funções que desempenham o processamento crítico fundamental do Sistema Robótico proposto, sua concepção, desenvolvimento e consolidação foi uma das partes que mais tempo consumiu tempo, logo atrás da pesquisa de fontes bibliográficas e de referencial teórico. Isto porque, se estas últimas tinham como propósito permitir encontrar a solução tecnológica que levaria aos resultados esperados (processamento da linguagem natural),

era a partir da interpretação prática desse conteúdo em si que as soluções tecnológicas surgiriam.

Como premissas básicas para o Sistema Robótico proposto, seriam utilizados um robô brasileiro (o Curumim), a linguagem de programação C# (já que é moderna, orientada a objetos, dispõe de uma vasta biblioteca de funções e recursos denominada *dotNet* e possui suporte tanto da sua empresa criadora, a *Microsoft*, quanto da comunidade de *software* livre – Mono), peças de *softwares* e algoritmos de programação aplicados ao processamento da linguagem natural desenvolvidos pela comunidade de *software* livre (sendo possível utilizá-los sem pagamento de *royalities*) e a linguagem natural oficial do Brasil (o Português).

Seu desenvolvimento ocorreu de acordo com as premissas básicas esclarecidas na seção 5.1 (Planejamento e restrição de contexto).

5.5.1 O PROCESSAMENTO SINTÁTICO

Foram cerca de 40 horas de trabalho de pesquisa em comunidades de *software* livre e fontes de informação disponíveis na internet para se chegar à biblioteca de funções denominada *SharpNLP*, que é uma portagem[6] para o C# (NORTHEDGE, 2006) da biblioteca de funções para processamento de linguagem natural denominada *OpenNLP* (BALDRIDGE; MORTON; BIERNER, 2000).

O *SharpNLP* parecia ser bastante promissor inicialmente, uma vez que dispunha de modelos treinados em Português como em outras línguas, é conceituado *software* de aplicação específica e gratuito. Alternativas comerciais foram também encontradas durante a pesquisa, porém não se mostraram atraentes em decorrência da questão financeira que contrariaria a premissa básica de utilizar *software* livre. Por ser a área do processamento da linguagem natural uma área do conhecimento em franco crescimento, novas opções de soluções deverão surgir após a publicação deste trabalho e optou-se por não mencionar nenhuma solução comercial encontrada.

Com o empenho das primeiras horas e semanas de trabalho de estudo para entender como o *SharpNLP* funcionava e como poderia ser útil no contexto do tema desta dissertação foi possível conceber e propor o rudimento visual da interface com o

[6] Portagem: termo técnico “portar um programa”, ou seja, traduzir de uma linguagem de programação para outra, incluindo a conversão de suas estruturas característica, formas de notação e definições conceituais.

usuário. Neste período também se devem incluir os estudos para conhecer e compreender a própria linguagem de programação C#, seus conceitos e convenções. Estes esforços ocorreram entre fevereiro e março de 2011.

Foi também durante estes esforços que se encontrou a primeira dificuldade importante, decorrente de uma limitação do modelo em língua portuguesa do Brasil e que precisava ser resolvido para que o *SharpNLP* apresentasse os resultados esperados. O *Corpus*, ou conjunto de grande quantidade de frases em linguagem natural previamente processados por mãos humanas, usado para treinar o Modelo de Entropia Máxima do *SharpNLP* é o Bosque (uma composição integralmente revisada dos *Corpora* CETENFolha e o CETEMPublico), ou seja, constituem-se de textos formais predominantemente jornalísticos onde as frases são frequentemente escritas com verbos na terceira pessoa e nos tempos Passado ou Presente (ocasionalmente no Futuro). Considerando que no âmbito de aplicação desta dissertação as instruções que seriam apresentadas ao sistema robótico estariam muito frequentemente no Imperativo, os primeiros testes revelaram que os verbos resultavam marcados sintaticamente incorretos como substantivos. Isto decorre do fato de que os verbos flexionados no Imperativo ocasionavam tanto palavras como contextos de uso inexistentes ou pouco comuns no *Corpus* de treinamento.

A solução correta para este problema seria dotar o *Corpus* de treinamento com frases no Imperativo. Tal trabalho seria virtualmente impossível de ser realizado dentro do contexto desta dissertação, já que requer milhares de horas de trabalho para processar sintaticamente frases manualmente, atribuindo classificações sintáticas palavra por palavra conforme seus usos em cada oração. O custo deste trabalho também estaria muito fora do escopo proposto.

O estudo da documentação técnica disponível para o *SharpNLP* revelou o recurso de um dicionário para palavras raras que seriam inseridas no modelo treinado resultante, com a marcação sintática correta. Nele, poderiam ser inseridas as palavras oriundas da flexão verbal no Imperativo conforme um planejamento prévio de situações de uso. Mesmo assim, optou-se por criar algumas dezenas de frases que foram acrescidas ao *Corpus* inicial de treinamento, com todas as palavras devidamente marcadas sintaticamente conforme as regras gramaticais e as marcações disponíveis no *Corpus*. A simples adição de algumas dezenas de frases planejadas no Corpus não permitiu resultado significativo no treinamento do modelo em Português do Brasil

resultado, já que se trata de Entropia Máxima e o universo de possibilidades de treinamento precisaria ser muito maior.

Essas frases adicionadas visariam auxiliar na resolução de ambiguidades de marcação. Por exemplo, a palavra "adiante" poderia ser usada por um usuário:

- Na frase "Eu quero que o robô **adiante** 1 metro" → verbo;
- Na frase "Eu quero que o robô siga **adiante** 1 metro" → advérbio.

O Modelo de Entropia Máxima do *SharpNLP* é capaz de avaliar trigramas (conjunto de três *tokens* ou palavras). No caso do exemplo, o dicionário apresenta duas marcações sintáticas diferentes para a palavra adiante (verbo e advérbio), ou seja, uma ambiguidade. Assim, as frases constituem em exemplos de uso e informação adicional aos algoritmos do *SharpNLP* para resolver esta ambiguidade.

Cabe ressaltar que o resultado apresentado pelo *SharpNLP* com essa solução paliativa nem sempre é como esperado. Em alguns casos ele não realizou a marcação correta das palavras e estatisticamente comete alguns erros e, felizmente, mais acertos. Uma sugestão de desenvolvimento futuro seria a elaboração de um *Corpus* com frases apropriadas ao contexto de uso de instruções em linguagem natural. Outra sugestão seria o estudo mais detalhado de como o algoritmo de Entropia Máxima do *SharpNLP* funciona quando utiliza o dicionário de palavras raras, uma vez que esse estudo detalhado está além do escopo deste trabalho.

Para a proposição de solução ao problema dos tempos verbais no imperativo foram empenhadas cerca de 80 horas de trabalho de pesquisa e um tempo proporcional de estudo, implementação, testes e ajustes: um total de cerca de 200 horas. Estes esforços ocorreram entre abril e junho de 2011.

5.5.2 O PROCESSAMENTO SEMÂNTICO

As primeiras etapas do processamento da linguagem natural consistem na separação dos períodos, na separação dos *tokens* ou palavras e na marcação sintática. Estas três etapas são realizadas pelo *SharpNLP*. As etapas seguintes requerem a interpretação do texto para estabelecer contexto de significado e extração de dados/informações. O *SharpNLP* não dispõe de recursos para tal.

Em retorno às pesquisas, para resolver o problema da interpretação do texto em linguagem natural deparou-se com as seguintes teorias, que suscitam métodos determinísticos de processamento:

- Gramática Semântica ou "*Semantic Grammar*" em inglês (BURTON; BROWN, 1977);
- Dependência Conceitual ou "*Conceptual Dependency*" em inglês (SCHANK, 1975); e
- Esquema de Casos ou "*Case Frames*" em inglês (FILLMORE, 1968a e 1968b).

Foram encontrados alguns estudos a respeito da marcação de funções semânticas em textos por métodos estocásticos (ARGILLANDER, 2005) (KONOPÍK, 2006), porém não foi possível obter detalhes suficientes de sua implementação para uso com o C# no Sistema Robótico proposto. Até o momento do encerramento desta fase (cerca de junho e julho de 2011), não foi possível encontrar nenhum *Corpus* de treinamento para marcação semântica disponível gratuitamente, muito menos em língua portuguesa do Brasil.

Assim, a pesquisa de referencial teórico resultou numa sugestão para métodos determinísticos como sendo praticamente mais promissores para o processamento semântico.

A Gramática Semântica assemelha-se à Gramática Livre de Contexto e a sua análise ("*parsing*") pode ser executada pelos mesmos algoritmos. Porém, esta análise seria bastante dependente de Elementos Terminais (palavras), o que significa um aumento rápido da complexidade da gramática quando se busca flexibilidade por meio de maior vocabulário e variedade na construção das frases. A língua portuguesa flexiona os verbos, gerando várias novas palavras que precisariam ser previstas no planejamento de tais gramáticas. Também, seria preciso definir formas de resolver as ambiguidades. Optou-se por não utilizar esta teoria.

A Dependência Contextual apresenta uma forma organizada de representar o significado de textos mais elaborados com uma história. Com um conjunto finito de significados elementares com possibilidade de representação que são usados de forma criativa. Porém, até o momento do encerramento das pesquisas sobre esta teoria, não foram encontrados referenciais teóricos nem fontes bibliográficas com detalhes suficientes de implementação de formas computacionais de marcação a partir de

linguagem natural que pudessem ser utilizados com o C#. Igualmente, optou-se por não utilizar esta teoria.

A teoria dos Esquemas de Casos proposta por Fillmore (1968a e 1968b) se mostrou interessante inicialmente por ser centrada nos verbos das orações, sendo que cada esquema de uso do verbo requer alguns elementos complementares opcionais ou obrigatórios. De fato, os verbos são elementos importantes no contexto de uso de instruções, como é o caso do Sistema Robótico proposto. Também, os verbos são palavras que aparecem de forma menos variada em frases de instruções (exceto pela conjugação típica da língua) e tendem a ser utilizados com mais cuidado, pois representam ações por si só e estão atreladas ao significado. Com essa teoria, seria possível iniciar a análise semântica a partir dos verbos das orações apresentadas pelo usuário e, com um conjunto de Esquemas de Casos previamente planejados e um dicionário de palavras-chaves, encontrar o esquema que mais provavelmente se ajustaria, para então utilizá-lo como referência para extração de dados e informações. Optou-se por utilizar os Esquemas de Casos para processamento semântico no Sistema Robótico.

Para a proposição de solução ao problema da análise semântica foram empenhadas cerca de 100 horas de trabalho de pesquisa e um tempo equivalente de estudo, implementação, testes e ajustes: um total de cerca de 350 horas. Estes esforços ocorreram entre abril e outubro de 2011.

5.5.3 PLANTA DO LOCAL E ALGORITMO DE BUSCA DE ROTA

Um ser humano, ao passar instruções para outro, naturalmente prefere fazer referência a pontos conhecidos ou que se chamam a atenção de alguma forma, para então agregar informações. Ou seja, em outras palavras, para um ser humano é mais comum e natural expressar instruções através de uma frase como:

- *Vá para perto da porta e fique à esquerda*.

E seria bem pouco natural expressar instruções através de uma frase como:

- *Levante do sofá, vire 90 graus para a direita, caminhe em frente cerca de 2 metros, vire 90 graus para a esquerda, avance 30 centímetros e pare*.

É mais simples e fácil comunicar uma instrução quando o interlocutor destinatário da mensagem dispõe de inteligência para resolver alguns problemas, tais como reconhecer as referências e ser capaz de resolver o problema da navegação até elas (buscar um caminho até lá). Assim, a capacidade de abstrair funções simples como movimentar e girar para usá-las de forma inteligente como buscar um caminho e percorrê-lo agregam bastante valor e utilidade a um sistema móvel.

Outro exemplo prático para ilustrar a importância desta forma de inteligência é o uso de serviço de taxi. Uma instrução ao taxista como "Siga ao aeroporto" é simples de ser comunicada. O taxista, de sua parte, possui conhecimento suficiente para saber o que é aeroporto, onde fica, quais as opções de caminho até lá em função do ponto em que pegou o passageiro e saberá, inclusive, o que fazer caso surjam imprevistos no caminho, como acidentes, por exemplo. Grande parte do valor deste serviço está no conhecimento do taxista e sua capacidade de buscar as soluções de trajeto e percurso.

A virtude desta forma de inteligência está na capacidade de simplificar a comunicação, porque basta dizer **o que** se deseja enquanto que a outra parte será capaz de expandir a ideia traduzindo-a em **como** realizar o que se deseja. No exemplo, não há necessidade de comunicar detalhadamente como executar o percurso, tarefa trabalhosa e geralmente entediante para seres humanos.

No exemplo do taxista, após comunicar o que deseja, o emissor da mensagem encontra-se livre para dedicar seu tempo em outras tarefas, já que o destinatário (o taxista) assumirá a missão de encontrar a solução para o problema. Há uma divisão de tarefas. E certamente o serviço, que tem seu valor, precisará ser devidamente remunerado.

Uma condição onde a divisão de tarefas se tornará uma solução necessária é quando o tempo para transmissão da mensagem é grande, havendo atraso considerável para seu recebimento no destinatário, para recebimento da resposta de volta ao emissor ou atraso nas duas vias. É o caso das explorações espaciais, em que as distâncias são enormes e os sinais podem demorar facilmente alguns minutos para se propagar entre os pontos comunicantes.

Assim, ao considerar que um robô móvel, movendo-se por um ambiente estranho em outro planeta, tal não poderá simplesmente ser "dirigido" a partir da Terra da mesma maneira como se dirige um carro. Se estiver rumando para buraco ou cratera,

o sinal de uma câmera fixada no robô precisaria de alguns minutos para viajar pelo espaço e ser exibida ao usuário do robô e o comando de correção de rota para evitar o acidente precisaria de igual quantidade de minutos para viajar de volta até o robô no planeta em exploração. Essa condição multiplica a probabilidade de riscos e pode facilmente ser catastrófica no caso de um erro de transmissão. Assim, uma solução possível seria dotar o robô ou o sistema local de inteligência suficiente para resolver o problema dos principais riscos de navegação. Neste caso, o comando da exploração espacial poderia simplesmente dizer para onde o robô deverá ir e ele (possivelmente com auxílio do sistema de suporte no local) seria capaz de traçar a rota de um ponto até outro e contornar os principais problemas e riscos de navegação. Um trabalho planejado de exploração em conjunto com o controle da missão minimizariam ainda mais os riscos e poderiam maximizar os resultados das oportunidades imprevistas.

Portanto, dotar um sistema robótico móvel com a capacidade de buscar por si só as rotas de navegação entre dois pontos pode ter várias utilidades e consistirá em interessante recurso para experimentos, estudos e desenvolvimentos.

Para dotar o sistema robótico proposto dos recursos necessários para permitir-lhe a capacidade de buscar caminhos, foram feitas inicialmente pesquisas de referenciais teóricos disponíveis. Neste ponto do desenvolvimento, o tempo disponível já se apresentava escasso. Então, era preciso uma solução simples, prática e rápida, que se constituísse em conteúdo tecnicamente interessante e compatível com o trabalho em andamento. Felizmente os algoritmos para este tipo de recurso foram desenvolvidos e são utilizados em jogos eletrônicos do tipo "*videogame*" há algumas décadas.

A pesquisa de referencial teórico na literatura especializada em inteligência artificial e na internet mostrou que se trata do uso de Algoritmos de Busca Heurística de Solução. A seção 2.8 (Algoritmo de busca do melhor caminho) comenta sobre estes algoritmos, porém, por razão de simplicidade e economia de tempo, optou-se por verificar somente as versões originais e não as variantes. Como sugestão de desenvolvimentos futuros, cabe o estudo da aplicação dessas variantes.

Para desenvolvimento deste recurso de *software* em C# para o sistema robótico proposto, independentemente do algoritmo escolhido, será necessário interagir com o usuário sobre o conteúdo de memória para formar uma matriz suficientemente detalhada que consistirá no mapa ou planta do ambiente onde o robô se moverá. Para determinar o tamanho dessa matriz, foram considerados os seguintes requisitos:

- Tamanho da própria matriz: quanto menor, menos complexa;
- Precisão para detalhes da planta resultante: quanto mais precisa, melhor;
- Tamanho da área total coberta pela planta; e
- Dimensões típicas do robô (cerca de 20 x 20 cm).

Por razões de simplificação, adotou-se como precisão típica da planta a própria dimensão do robô Curumim escolhido para o sistema robótico proposto, ou seja, quadrados de 20 x 20 cm. Porém, a escala pode ser ajustada pelo usuário, conforme a necessidade do experimento. O tamanho prático da matriz acabou resultando em 15 x 15 quadrados, pois é um número decimal fácil de usar e, ao se posicionar o ponto de origem (zero) no centro da matriz, será possível encontrar um número igual de 7 quadrados em qualquer um dos sentidos cartesianos até a borda da planta. Como o registrador interno da posição do robô, no sistema, possui precisão em unidades de milímetro, um controle mais preciso não será comprometido. Ao lado disso, há problemas de escorregamento das rodas e questões de imprecisão do movimento que superam, em muito, a precisão desses registradores. Também, com 15 quadrados de 20 cm cada, será possível mapear uma área de 3 x 3 metros, parecendo ser suficiente para a maioria dos experimentos práticos de pequena escala e distância segura para manter o alcance da comunicação sem fio via rádio do Curumim. A experiência mostrou que o alcance da comunicação sem fio diminui perto do final da carga da bateria, após algumas horas de uso. Igualmente, a experiência mostrou que a precisão escolhida para a planta é suficiente para, inclusive, evidenciar a imprecisão do movimento do robô. O Apêndice A.4 apresenta mais sobre um teste da comunicação sem fio.

A forma de utilização do algoritmo de busca de rota é simples. Trata-se de uma função de *software*, sendo que a variável de entrada é a matriz da planta e a variável de saída é a matriz da planta preenchida com a melhor rota identificada. Cada elemento da matriz da planta possui um código indicativo para representar uma barreira, um caminho livre, o ponto de origem, o ponto de destino ou um ponto pertencente à rota. Porém, a matriz interna do sistema robótico proposto possui outros códigos indicativos, para marcar pontos de referência úteis. Assim, antes do sistema originar uma chamada à função de busca de rota, é preciso preencher variável com uma matriz filtrando-se os pontos de referência excedentes, ou seja, que não serão usados nem como origem (posição atual do robô) nem como destino. Para encontrar o caminho, o sistema vasculha a matriz de saída, extraindo os pontos relativos à rota e organizando-os em

ordem de precedência, da origem ao destino. Uma vez dispondo da sequência de pontos pertencentes à rota, os procedimentos de Lógica e Controle do Robô realizam a transcrição para uma sequência apropriada de comandos de movimento e rotação, com seus respectivos parâmetros.

A funcionalidade básica da Planta do local é decorrente de adaptações e ajustes feitos no código em C# original de Allen (2008). Apensar de fornecer um algoritmo de busca de rota, este não pode ser aproveitado, uma vez que utiliza força bruta para buscar a solução. Ao invés disso, foi adaptado o algoritmo A* (A-Estrela) proposto por Husse (2010), por se tratar de implementação computacionalmente rápida e que poderá ser utilizada futuramente, inclusive, em plantas bem mais complexas ou com resolução muito superior.

Para a proposição de solução ao problema do algoritmo de busca de rota foram empenhadas cerca de 40 horas de trabalho de pesquisa. Somando-se o tempo necessário para estudo, implementação, testes e ajustes, o total até o ponto de um resultado utilizável somou cerca de 100 horas. Estes esforços ocorreram entre julho e outubro de 2011.

6. TESTES E RESULTADOS

Nesta seção, que consiste de duas partes, serão apresentados os testes realizados com o sistema robótico proposto e os resultados obtidos.

Na primeira parte, são apresentados os experimentos formais realizados com usuários reais, ou seja, usuários em potencial do sistema, leigos a respeito de robótica e inteligência artificial. Objetivam levantar informações a respeito das impressões desses usuários sobre as funcionalidades e o desempenho do sistema.

Na segunda parte, são apresentados os testes realizados no sistema robótico. São testes realizados objetivando não só avaliar o desempenho e a funcionalidade do trabalho desenvolvido até o momento em que os testes foram realizados, como também documentar de forma mais clara o funcionamento, as possibilidades, as limitações e as melhorias que podem ser sugeridas e agregadas no futuro.

O método utilizado nesta segunda parte é intuitivo, ou seja, as instruções que o usuário apresentará serão escritas em Português, em linguagem natural, e o resultado da interpretação do texto será exibido na Interface em Linguagem Natural. Ficará à escolha do usuário executar no robô a sequência de comandos obtida ou reescrever a instrução, dependendo dos objetivos pretendidos.

6.1 TESTES DO SISTEMA ROBÓTICO PROPOSTO COM USUÁRIOS LEIGOS

Os testes e resultados a seguir foram obtidos na última revisão do sistema robótico proposto, ou seja, a versão mais desenvolvida.

No Sistema Robótico Proposto foi acrescentado recurso que permite registrar automaticamente os dados contidos em um conjunto de variáveis de forma a permitir avaliar seu desempenho na interpretação das instruções em linguagem natural.

Para tanto, foi proposto um experimento padronizado a ser aplicado a usuários reais, dividido em duas partes.

A Primeira Parte permitirá ao usuário experimentar livremente o sistema e, assim, familiarizar-se com seus recursos, possibilidades e funcionamento. A Segunda

Parte consiste em uma tarefa com objetivo previamente determinado, que permitirá avaliar a eficiência/eficácia do sistema com relação aos diferentes usuários.

Ambas as partes se apresentam com limite de tempo preestabelecido, sendo que o usuário poderá optar por mais tempo se desejar, ou seja:

- Primeira parte: dez (10) minutos iniciais e mais duas prorrogações de dez (10) minutos cada, totalizando trinta (30) minutos máximos; e
- Segunda parte: vinte e cinco (25) minutos inicias, uma prorrogação de quinze (15) minutos e outra de dez (10) minutos, totalizando cinquenta (50) minutos máximos.

Portanto, o experimento prevê uma duração não maior que oitenta (80) minutos, ou seja, uma hora e vinte, buscando evitar a influência negativa do fator cansaço do usuário, que adicionaria complexidade desnecessária. Também, prevê um tempo mínimo de trinta e cinco (35) minutos para tomar familiaridade e executar a tarefa de forma consistente o suficiente para a comparação pretendida.

6.1.1 O AMBIENTE DA EXPERIÊNCIA

Consiste em local com espaço livre suficiente para a movimentação do robô, um computador com o Sistema Robótico instalado e funcionando e acomodações para receber o usuário confortavelmente sentado.

Uma vez que relatórios precisam ser preenchidos antes e depois da experiência, acomodações apropriadas foram providenciadas (mesa, caneta etc).

6.1.2 PADRONIZAÇÃO DA EXPERIÊNCIA

A padronização permitirá realizar comparações entre os resultados perante um mesmo referencial. Mais detalhes sobre a padronização adotada pode ser vista no Apêndice A.2.

A experiência foi conduzida com cinco (5) usuários diferentes, que não se repetiram durante a realização. Para coletar os dados e informações foram utilizados: questionários, registro automáticos de eventos do sistema robótico e anotações feitas pelo pesquisador durante a realização dos experimentos.

O experimento se constitui de cinco passos, a saber:

- Passo 01: apresentação padronizada da experiência e introdução à Interface em Linguagem Natural;
- Passo 02: leitura do "Descritivo da Experiência", preenchimento do "Termo de Esclarecimento e Livre Consentimento" e do "Questionário Demográfico";
- Passo 03: realização da Primeira Parte da experiência, permitindo que o usuário familiarize-se com sistema;
- Passo 04: realização da Segunda Parte da experiência, permitindo registros de avaliação do desempenho frente a um objetivo estabelecido; e
- Passo 05: terminada a experiência, preenchimento do "Questionário sobre a experiência de uso".

Como finalização, foi feita consulta informal aos usuários, para coletar possíveis comentários ou observações últimos que poderiam trazer melhores conhecimentos sobre a interação.

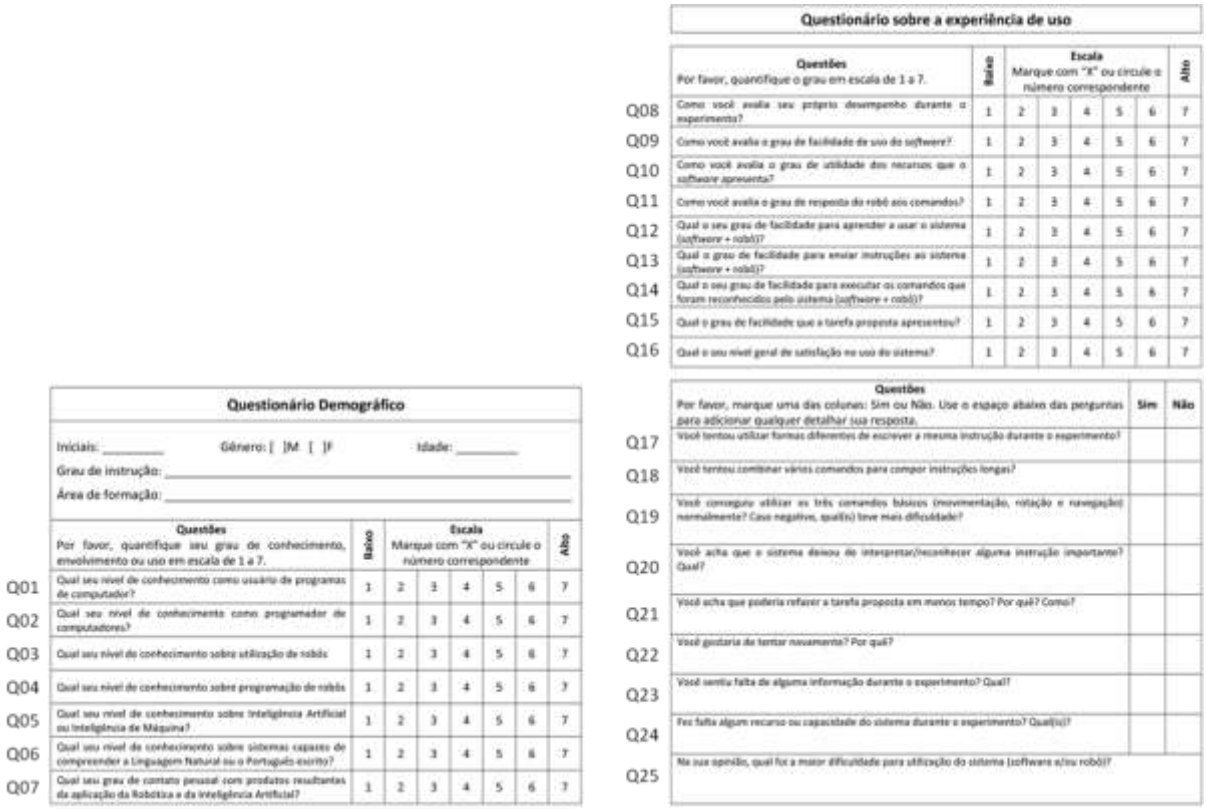
6.1.3 APRESENTAÇÃO E AVALIAÇÃO DOS RESULTADOS

Questionários preenchidos pelos usuários

Para iniciar a apresentação e a avaliação dos resultados, foram tabulados as respostas dos questionários e os dados registrados automaticamente pelo sistema, considerando as duas partes da experiência.

De forma a facilitar a identificação das questões, elas foram numeradas de Q1 a Q25. A figura 6.1.1 indica esta numeração.

O propósito do Questionário Demográfico é registrar algumas informações sobre o usuário e seus conhecimentos, para permitir uma comparação com os resultados durante a análise; o do Questionário sobre a experiência de uso é registrar aspectos sobre a interação entre o usuário e o sistema robótico proposto por meio da interface em linguagem natural.



Questionário Demográfico

Iniciais: _____ Gênero: [ ]M [ ]F Idade: _____

Grau de instrução: _____

Área de formação: _____

	Questões Por favor, quantifique seu grau de conhecimento, envolvimento ou uso em escala de 1 a 7.	Baixo	Escala Marque com "X" ou circule o número correspondente					Alto
Q01	Qual seu nível de conhecimento como usuário de programas de computador?	1	2	3	4	5	6	7
Q02	Qual seu nível de conhecimento como programador de computadores?	1	2	3	4	5	6	7
Q03	Qual seu nível de conhecimento sobre utilização de robôs	1	2	3	4	5	6	7
Q04	Qual seu nível de conhecimento sobre programação de robôs	1	2	3	4	5	6	7
Q05	Qual seu nível de conhecimento sobre Inteligência Artificial ou Inteligência de Máquina?	1	2	3	4	5	6	7
Q06	Qual seu nível de conhecimento sobre sistemas capazes de compreender a Linguagem Natural ou o Português escrito?	1	2	3	4	5	6	7
Q07	Qual seu grau de contato pessoal com produtos resultantes da aplicação da Robótica e da Inteligência Artificial?	1	2	3	4	5	6	7

Questionário sobre a experiência de uso

	Questões Por favor, quantifique o grau em escala de 1 a 7.	Baixo	Escala Marque com "X" ou circule o número correspondente					Alto
Q08	Como você avalia seu próprio desempenho durante o experimento?	1	2	3	4	5	6	7
Q09	Como você avalia o grau de facilidade de uso do *software*?	1	2	3	4	5	6	7
Q10	Como você avalia o grau de utilidade dos recursos que o *software* apresenta?	1	2	3	4	5	6	7
Q11	Como você avalia o grau de resposta do robô aos comandos?	1	2	3	4	5	6	7
Q12	Qual o seu grau de facilidade para aprender a usar o sistema (*software* + robô)?	1	2	3	4	5	6	7
Q13	Qual o grau de facilidade para enviar instruções ao sistema (*software* + robô)?	1	2	3	4	5	6	7
Q14	Qual o seu grau de facilidade para executar os comandos que foram reconhecidos pelo sistema (*software* + robô)?	1	2	3	4	5	6	7
Q15	Qual o grau de facilidade que a tarefa proposta apresentou?	1	2	3	4	5	6	7
Q16	Qual o seu nível geral de satisfação no uso do sistema?	1	2	3	4	5	6	7

	Questões Por favor, marque uma das colunas: Sim ou Não. Use o espaço abaixo das perguntas para adicionar qualquer detalhar sua resposta.	Sim	Não
Q17	Você tentou utilizar formas diferentes de escrever a mesma instrução durante o experimento?		
Q18	Você tentou combinar vários comandos para compor instruções longas?		
Q19	Você conseguiu utilizar os três comandos básicos (movimentação, rotação e navegação) normalmente? Caso negativo, qual(is) teve mais dificuldade?		
Q20	Você acha que o sistema deixou de interpretar/reconhecer alguma instrução importante? Qual?		
Q21	Você acha que poderia refazer a tarefa proposta em menos tempo? Por quê? Como?		
Q22	Você gostaria de tentar novamente? Por quê?		
Q23	Você sentiu falta de alguma informação durante o experimento? Qual?		
Q24	Fez falta algum recurso ou capacidade do sistema durante o experimento? Qual(is)?		
Q25	Na sua opinião, qual foi a maior dificuldade para utilização do sistema (software e/ou robô)?		

Figura 6.1.1 – Questionários aplicados e questões numeradas de Q1 a Q25.

A tabela 6.1.1 agrupa as respostas encontradas para ambos os questionários.

Tabela 6.1.1 – Tabulação das respostas às questões da figura 6.1.1.

Questão	Usuário 1	Usuário 2	Usuário 3	Usuário 4	Usuário 5
Q01	5	6	4	2	7
Q02	1	2	1	1	7
Q03	1	3	1	1	2
Q04	1	1	1	1	1
Q05	1	1	1	1	2
Q06	1	1	1	1	2
Q07	1	1	1	1	1
Q08	4	5	4	4	6
Q09	3	6	5	5	6
Q10	5	6	5	5	6
Q11	3	4	4	4	4
Q12	4	6	5	4	6
Q13	6	6	6	6	5
Q14	7	7	7	7	6
Q15	5	6	5	5	5
Q16	5	6	6	7	5

Tabela 6.1.1 (continuação) – Tabulação das respostas às questões da figura 6.1.1.

Questão	Usuário 1	Usuário 2	Usuário 3	Usuário 4	Usuário 5
Q17	Sim	Sim	Sim	Sim	Sim
Q18	Não	Sim	Sim	Sim	Sim
Q19	Sim. Não consegui fazer o robô rodar direito as primeiras vezes.	Sim	Sim	Sim	Sim
Q20	Sim. O robô não rodou direito algumas vezes.	Sim. Não compreendeu como ir para o pontosc depois de ir para frente e virar para a esquerda.	Sim. Não reconheceu o comando de fazer rotação total e de movimentação conjugada.	Sim. Não reconheceu o comando de girar para a direita 180°.	Sim. Mudança na ordem nos comandos, negativas, movimentos combinados.
Q21	Sim. Talvez dê para fazer mais fácil agora que sei como comandar o robô.	Sim. Já tenho mais prática.	Sim. Agora já sei usar melhor o sistema.	Sim. Já adquiri mais prática.	Sim. Com um pouco mais de prática, ganhando mais agilidade.
Q22	Não.	Sim. Testar um pouco mais.	Sim. Achei interessante.	Sim. É divertido. Fazer outras coisas.	Sim. Para tentar outras coisas.
Q23	Não.	Não.	Não. O entrevistador esclareceu minhas dúvidas.	Não	Não
Q24	Sim. Melhor controle do robô.	Não.	Sim. Poder buscar um comando que digitei anteriormente.	Sim. A comunicação com o robô é imprecisa.	Sim. A comunicação com o robô é imprecisa.
Q25	Dificuldade de mandar o robô fazer o que eu queria. Às vezes não entende, às vezes não faz.	Comandos não reconhecidos. Robô pára às vezes.	Alguns comandos não foram reconhecidos e o robô às vezes trava.	Falta de prática com o computador e algumas palavras que não foram reconhecidas.	Algumas palavras ou expressão não foram reconhecidas e a comunicação com o robô é imprecisa.
Idade	32	40	38	72	38
Gênero	F	M	F	F	M
Instrução	Superior Compl.	Superior Compl.	Superior Compl.	Superior Compl.	Superior Compl.
Área	Pedagogia	Eng. Mecânica (automotiva)	Desenho Industrial	Pedagogia	Informática

A tabela 6.1.2 apresenta o cálculo da Média, Mediana e Desvio Padrão para as respostas do Questionário Demográfico (Q1 a Q7).

Tabela 6.1.2 – Cálculos estatísticos sobre o Questionário Demográfico.

Questão	Média	Mediana	Desvio Padrão
Q1	4,8	5,0	1,7
Q2	2,4	1,0	2,3
Q3	1,6	1,0	0,8
Q4	1,0	1,0	0
Q5	1,2	1,0	0,4
Q6	1,2	1,0	0,4
Q7	1,0	1,0	0

A partir do Questionário Demográfico observa-se que quase todos os usuários participantes não tem conhecimento sobre Robótica (Q3, Q4 e Q7), Inteligência Artificial (Q5 e Q7), Processamento da Linguagem Natural (Q6 e Q7) e programação de computadores (Q2), porém a maioria tem conhecimento como usuário de computadores (Q1). Um dos usuários, com formação em informática, tem bom conhecimento de programação e, tanto ele como outro usuário, engenheiro, tem algum conhecimento sobre robótica.

A tabela 6.1.3 apresenta o cálculo da Média, Media e Desvio Padrão para o Questionário sobre a experiência de uso (Q08 a Q16).

Tabela 6.1.3 – Cálculos estatísticos para o Questionário sobre a experiência de uso.

Questão	Média	Mediana	Desvio Padrão
Q08	4,6	4,0	0,8
Q09	5,0	5,0	1,1
Q10	5,4	5,0	0,5
Q11	3,8	4,0	0,4
Q12	5,0	5,0	0,9
Q13	5,8	6,0	0,4
Q14	6,8	7,0	0,4
Q15	5,2	5,0	0,4
Q16	5,8	6,0	0,7

De maneira geral, os usuários que experimentaram o sistema se sentiram capazes de operá-lo satisfatoriamente e nenhum considerou seu próprio desempenho

abaixo de um desempenho mediano (Q08). Um usuário considerou seu próprio desempenho como quase ótimo (nota 6,0).

Na questão sobre a facilidade de uso da Interface em Linguagem Natural e o *software* do Sistema Robótico (Q09), apenas um usuário o avaliou como abaixo de um desempenho mediano, porém a maioria o considerou bom, resultando em avaliação média e mediana iguais a cinco (5,0).

Sobre o grau de utilidade dos recursos disponibilizados pela Interface em Linguagem Natural e o *software* do Sistema Robótico (Q10), igualmente a maioria dos usuários o considerou bom, resultando em avaliação média acima de cinco (5,4) e mediana cinco (5,0). Nenhum usuário reportou avaliação ruim (nota menor que 4,0) para a utilidade dos recursos disponibilizados.

O grau de resposta do robô (Q11) foi o item com avaliação mais baixa na experiência de uso, segundo os usuários pesquisados, atingindo pontuação mediana de quatro (4,0). As respostas para Q24 e Q25 completam este resultado. É preciso notar as observações feitas ao longo deste trabalho sobre a comunicação sem fio, uma vez o robô possui constituição bastante robusta e não apresentou nenhum problema.

A maioria dos usuários considerou-se satisfeita com relação à capacidade de aprender a usar a Interface em Linguagem Natural e o Sistema Robótico (Q12). Apenas dois usuários consideraram-se com desempenho mediano (4,0), sendo que a média e a mediana da avaliação resultaram em valor igual a cinco (5,0). A avaliação deste quesito pode ser completada pelas respostas às questões Q21 (refazer a tarefa em menos tempo) e Q22 (tentar novamente), pois todos manifestaram que se acreditam capazes de refazer a tarefa mais rapidamente e apenas um não gostaria de tentar novamente.

Ao se perguntar sobre a facilidade para enviar instruções ao sistema (Q13), deve-se considerar os dois significados desta questão. O primeiro se refere ao funcionamento da Interface em Linguagem Natural e o segundo se refere à capacidade do Sistema Robótico em interpretar as instruções fornecidas. Analisando os resultados das questões Q17, Q18, Q19 e Q20 é possível conhecer a impressão dos usuários sobre a capacidade do sistema em interpretar as instruções.

De maneira geral, os usuários que experimentaram o sistema relataram estarem satisfeitos com a facilidade de uso da Interface em Linguagem Natural ou com a explicação de uso que receberam do pesquisador. A avaliação resultou em média superior a cinco (5,8). Porém, a avaliação sobre a capacidade de interpretação do

sistema das tentativas de comandar o robô resultou em considerações por parte de todos os usuários.

Todos os usuários consideraram bem fácil disparar a execução dos comandos interpretados (Q14) e esta questão resultou em avaliação média superior a seis (6,8).

A tarefa proposta para a segunda parte do experimento também foi considerada bem fácil de ser executada através do sistema (Q15). Recebeu avaliação média superior a cinco (5.2).

Houve satisfação geral com a experiência de utilização do Sistema Robótico, em vista da avaliação média superior a cinco (5,8) e da baixa dispersão entre a opinião dos usuários pesquisados (desvio padrão 0,4).

As principais considerações dos usuários concentram-se em torno da dificuldade do sistema interpretar as instruções fornecidas, o que implica em possibilidade de mais desenvolvimento para o processamento da linguagem natural.

Registros automáticos do sistema

Com relação aos dados registrados automaticamente pelo sistema, a partir do processamento destes registros, oriundos das variáveis do sistema, foi possível propor alguns parâmetros de comparação. Uma vez que cada usuário comportou-se de forma a produzir quantidades diferentes de instruções, os valores foram normalizados em relação a este dado. Os parâmetros são os seguintes:

- Tempo Total: quantidade de tempo utilizada pelo usuário durante a parte do experimento, na notação “horas:minutos:segundos”;
- Tempo entre Instruções: intervalo médio utilizado pelo usuário para inserir (digital) suas instruções na Interface em Linguagem Natural, na notação “horas:minutos:segundos”;
- Palavras/Instrução: número médio de palavras contidas nas instruções fornecidas pelo usuário ao sistema;
- Sentenças/Instrução: número médio de sentenças ou orações contidas nas instruções fornecidas pelo usuário ao sistema;
- Tentativas/Instrução: indica quantas tentativas adicionais o usuário fez, na média, para conseguir que uma mesma instrução fosse interpretada

pelo sistema (um valor igual a um significa que teve que tentar duas vezes cada instrução – uma errada e uma certa);

- Erros/Instrução (usuário ou escopo): proporção de erros por instrução atribuídos ao usuário (grafia, gramática, notação etc.) ou atribuídos à sua forma de expressão cuja compreensão está além do escopo deste trabalho (limitação, instruções impossíveis de serem interpretadas pelo sistema);
- Erros/Instrução (Proc. Semântico): proporção de erros por instrução atribuídos ao sistema e relativo ao processamento semântico;
- Erros/Instrução (Proc. Sintático): proporção de erros por instrução atribuídos ao sistema e relativo ao processamento sintático;
- Erros/Instrução (Sep. Palavras): proporção de erros por instrução atribuídos ao sistema e relativo à separação de palavras/*tokens* em cada sentença ou oração;
- Erros/Instrução (Sep. Sentenças): proporção de erros por instrução atribuídos ao sistema e relativo à separação das sentenças em cada instrução; e
- Erros/Instrução (Totais): proporção do total de erros por instrução atribuídos ao sistema, ou seja, a soma dos erros de separação de sentenças, separação de palavras, processamento sintático e processamento semântico.

Nota: todos os parâmetros denominados “Erros” significam que foi impossível realizar a interpretação correta da sentença (parte da instrução).

Primeira parte da experiência

A tabela 6.1.4 exibe extrato do conteúdo disponível nos registros automáticos para a Primeira Parte da experiência com usuários.

A tabela 6.1.5 apresenta o cálculo da Média, Mediana e Desvio Padrão para os resultados contidos na tabela 6.1.4.

Tabela 6.1.4 – Extrato dos dados da Primeira Parte da experiência.

Parâmetro	Usuário 1	Usuário 2	Usuário 3	Usuário 4	Usuário 5
Erros/Instrução (Totais do sistema)	26,7%	22,7%	17,6%	46,2%	12,5%
Erros/Instrução (Sep. Sentenças)	0,0%	0,0%	0,0%	0,0%	0,0%
Erros/Instrução (Sep. Palavras)	0,0%	0,0%	0,0%	23,1%	0,0%
Erros/Instrução (Proc. Sintático)	13,3%	9,1%	5,9%	7,7%	0,0%
Erros/Instrução (Proc. Semântico)	13,3%	13,6%	11,8%	15,4%	12,5%
Erros/Instrução (Usuário ou Escopo)	53,3%	27,3%	52,9%	61,5%	29,2%
Tentativas/Instrução	0,47	0,27	0,47	0,62	0,17
Sentenças/Instrução	1,00	1,00	1,12	1,08	1,04
Palavras/Instrução	4,47	2,82	4,35	5,54	4,38
Tempo entre Instruções (min.)	00:01:44	00:01:08	00:01:30	00:02:06	00:00:28
Tempo Total (min.)	00:25:58	00:24:51	00:25:25	00:27:12	00:11:04

Tabela 6.1.5 – Cálculos estatísticos dos resultados tabela 6.1.4.

Parâmetro	Média	Mediana	Desvio Padrão
Erros/Instrução (Totais do sistema)	25,14%	22,73%	11,54%
Erros/Instrução (Sep. Sentenças)	0,00%	0,00%	0,00%
Erros/Instrução (Sep. Palavras)	4,62%	0,00%	9,23%
Erros/Instrução (Proc. Sintático)	11,82%	9,09%	10,43%
Erros/Instrução (Proc. Semântico)	13,32%	13,33%	1,22%
Erros/Instrução (Usuário ou Escopo)	44,85%	52,94%	13,93%
Tentativas/Instrução	0,40	0,47	0,16
Sentenças/Instrução	1,05	1,04	0,05
Palavras/Instrução	4,31	4,38	0,87
Tempo entre Instruções (hh:mm:ss)	00:01:23	00:01:30	00:00:33
Tempo Total (hh:mm:ss)	00:22:54	00:25:25	00:05:58

Para a primeira parte do experimento, ao analisar os parâmetros de erro por instrução, nota-se que praticamente a metade dos erros observados se refere a erros do usuário ou de sua forma de redação que o sistema não é capaz de interpretar. Em seguida, aparecem os erros no processamento semântico (interpretação) e depois os erros no processamento sintático (classificação gramatical incorreta). Não foram observados erros de separação de sentenças durante o experimento com esses usuários.

Os usuários 2 e 5 são do gênero masculino e nota-se que a proporção de erros relativos ao usuário ou à sua forma de expressão incompreensível pelo sistema (além do escopo) é menor do que os usuários do gênero feminino. Também, parece haver leve

tendência de mulheres usarem mais palavras do que homens para compor instruções (Palavras/Instrução).

Da mesma forma, parece haver leve tendência dos homens usarem menos tempo para escrever suas instruções (tempo entre instruções).

Praticamente todos preferiram utilizar instruções com somente uma sentença a maior parte do tempo, o que resulta na interpretação de um único comando por instrução, por parte do sistema.

O tempo utilizado pelos usuários para executarem a experiência foi parecido, excedo o usuário 5, com mais conhecimento em informática, que preferiu terminar a primeira parte mais rapidamente.

As tentativas para conseguirem a interpretação correta de comandos foram em quantidade menor do que um, mas bem próximo de 0,5 na média. Ou seja, praticamente para cada duas instruções, uma delas precisou ser reescrita.

Segunda parte da experiência

A tabela 6.1.6 exibe extrato do conteúdo disponível nos registros automáticos para a Segunda Parte da experiência com usuários.

Tabela 6.1.6 – Extrato dos dados da Segunda Parte da experiência.

Parâmetros	Usuário 1	Usuário 2	Usuário 3	Usuário 4	Usuário 5
Erros/Instrução (Totais)	10,5%	9,1%	11,8%	10,5%	14,3%
Erros/Instrução (Sep. Sentenças)	0,0%	0,0%	0,0%	0,0%	0,0%
Erros/Instrução (Sep. Palavras)	0,0%	0,0%	0,0%	0,0%	0,0%
Erros/Instrução (Proc. Sintático)	0,0%	9,1%	5,9%	0,0%	14,3%
Erros/Instrução (Proc. Semântico)	10,5%	0,0%	5,9%	10,5%	0,0%
Erros/Instrução (Usuário ou Escopo)	5,3%	9,1%	5,9%	0,0%	0,0%
Tentativas/Instrução	0,11	0,18	0,12	0,11	0,14
Sentenças/Instrução	1,00	1,00	1,12	1,00	1,43
Palavras/Instrução	5,16	4,45	3,47	5,11	6,00
Tempo entre Instruções (hh:mm:ss)	00:01:20	00:01:20	00:01:22	00:01:37	00:00:21
Tempo Total (hh:mm:ss)	00:25:28	00:14:42	00:23:19	00:30:42	00:02:29

A tabela 6.1.7 apresenta o cálculo da Média, Mediana e Desvio Padrão para os resultados contidos na tabela 6.1.6.

Tabela 6.1.7 – Cálculos estatísticos dos resultados tabela 6.1.6.

Parâmetros	Média	Mediana	Desvio Padrão
Erros/Instrução (Totais)	11,24%	10,53%	1,74%
Erros/Instrução (Sep. Sentenças)	0,00%	0,00%	0,00%
Erros/Instrução (Sep. Palavras)	0,00%	0,00%	0,00%
Erros/Instrução (Proc. Sintático)	5,85%	5,88%	5,48%
Erros/Instrução (Proc. Semântico)	5,39%	5,88%	4,71%
Erros/Instrução (Usuário ou Escopo)	4,05%	5,26%	3,55%
Tentativas/Instrução	0,13	0,12	0,03
Sentenças/Instrução	1,11	1,00	0,17
Palavras/Instrução	4,84	5,11	0,84
Tempo entre Instruções (hh:mm:ss)	00:01:12	00:01:20	00:00:26
Tempo Total (hh:mm:ss)	00:19:20	00:23:19	00:09:53

Na segunda parte do experimento, observou-se menor proporção de erros relativos ao usuário ou sua forma de expressão incompreensível pelo sistema (além do escopo), porém, ainda foram observados erros no processamento sintático e semântico. Isto pode ser decorrente da prática e aprendizado adquiridos pelos usuários como também da pressão para concluir a tarefa especificada.

Observou-se também diminuição nas tentativas para conseguir a interpretação de comandos, que na segunda parte apresentou-se com valor pouco maior que 0,1, ou seja, para cada dez instruções, uma precisou ser reescrita. Isto sugere o aprendizado por parte do usuário sobre o vocabulário compreensível pelo sistema e as formas de expressão que pode processar.

Da mesma forma que na primeira parte, não foram observados erros do sistema para separação de sentenças. Também não foram observados erros de para separação de palavras na segunda parte do experimento.

Os usuários prosseguiram preferindo enviar instruções com uma única sentença (ou comando), exceto pelo usuário 6, que aumentou a proporção de instruções com mais de uma sentença.

Houve uma mudança de comportamento entre os usuários no que se refere ao número de palavras por instrução. A maioria preferiu utilizar instruções com mais palavras, especialmente o usuário 6, exceto o usuário 3, que diminuiu ao utilizar instruções com menos palavras.

Com relação ao tempo utilizado para concluir a tarefa, a maioria conseguiu dentro do primeiro prazo fixado, especialmente o usuário 6. Porém, o usuário 4, que o

ultrapassou, conseguiu concluir com pouco mais de cinco minutos adicionais. Vale observar que se trata de pessoa com pouca familiaridade com computadores e apresentou os maiores tempos entre instruções dentre os usuários que participaram do experimento.

6.1.4 OBSERVAÇÕES DO PESQUISADOR

Como observação do pesquisador que conduziu os experimentos, cabe notar que os usuários geralmente demonstraram sentir estranhamento inicial com a capacidade de interpretação das instruções digitadas por parte do sistema. Porém, à medida que prosseguiram experimentando e observando erros e acertos, houve uma adaptação ou acomodação no estilo de redação.

Na primeira parte, o usuário 4 demonstrou frustração ao tentar por várias vezes descrever detalhadamente o que desejava executar no robô, sem obter o resultado: “Faça uma rotação completa para a direita,180 graus”. Reescreveu a instrução várias vezes usando verbos sinônimos até descobrir os verbos “girar” e “virar”, incluídos no léxico. Também, por ter aproveitado o texto escrito inicialmente para reescrever, cometeu um erro ao não colocar espaço após a vírgula, antes do número “180”. Uma vez que no corpus de treinamento do Modelo de Entropia Máxima o mais comum é encontrar vírgula antes de número, isto fez com que o sistema cometesse um erro de separação de palavras/*tokens*, que se preferiu tabular e analisar. Poderia não ter sido computado como erro do sistema, uma vez que se trata de limitação, mas de erro do usuário. A separação do parâmetro 180 graus com vírgula não é interpretada como correta pelo sistema, que executa uma separação nesta pontuação e considera o segundo trecho como outra sentença. Como solicitou ajuda ao pesquisador, foi esclarecida que deveria colocar cada comando completamente entre sinais de pontuação e, por fim, conseguiu prosseguir com êxito. Depois, ela desabafou dizendo ter pensado em fazer o robô dançar.

Notou-se a tendência dos usuários preferirem comandos únicos, simples, com menos palavras e avaliarem logo em seguida o resultado por parte do robô. É possível que à medida que se sintam mais confiantes sobre a utilização do sistema e os resultados possíveis com o robô, optem por instruções elaboradas com mais comandos. O usuário familiarizado com informática comportou-se diferente dos demais neste aspecto. Da mesma forma, o pesquisador que desenvolveu o sistema é bastante familiarizado com o

sistema e escreve instruções mais elaboradas, como será mostrado a seguir na seção 6.2 (Testes das funcionalidades do sistema robótico proposto).

Conclui-se que o desempenho geral do sistema foi bom e conforme desejado.

6.2 TESTES DAS FUNCIONALIDADES DO SISTEMA ROBÓTICO PROPOSTO

Os testes a seguir objetivam avaliar funcionalidades, suas características e limitações do sistema, bem como problemas linguísticos ou possibilidades de desenvolvimento que poderão ser abordados em trabalhos futuros. Serão apresentados conforme o caso.

6.2.1 TESTE 1 – REQUISIÇÕES NA FORMA AFIRMATIVA

Do ponto de vista humano, quando se pretende dar uma instrução ou comando para alguém, a forma verbal normalmente utilizada é o imperativo, ou seja, frases do tipo:

- "Venha mais para perto da janela para ver o evento.";
- "Por favor, entre na sala."; e
- "Para chegar à loja, siga em frente até final da rua, vire à direita e continue...".

Porém, há outras formas de se transmitir instruções através de frases afirmativas, também usuais no cotidiano humano e geralmente menos associadas a instruções de comando. São sentenças construídas como, entre outras maneiras, uma espécie de confissão, compartilhamento de informação ou conselho, em frases do tipo:

- "O manual diz que este objeto deve ficar na entrada.";
- "Avisaram que o chefe pediu para que você fosse à garagem.";
- "Acho melhor você ir para a saída."; e
- "Talvez fosse melhor ficar mais para a esquerda."

Na estrutura superficial, ou seja, na construção das sentenças com recursos léxicos e sintáticos, são todas bastante diferentes entre si e cada uma sugere um contexto diferente de utilização. Sem dúvida sinalizam que há outros acontecimentos

relacionados, anteriores e posteriores, importantes aos Seres Humanos envolvidos. Há várias camadas de significados em cada uma. Contudo, na estrutura profunda, é possível dizer que todas expressam basicamente uma mesma ideia, um deslocamento de algo ou alguém de um lugar para outro, ao se considerar a camada de significado relacionada à ação de movimento.

Serão apresentados nesta seção os testes realizados com instruções construídas por meio de afirmativas e os resultados apresentados pelo Sistema Robótico proposto, pela sequência de comandos.

Como teste de instruções na forma imperativa com as ações explicitadas, ou seja, as movimentações e rotações foram listadas em sequência, uma por vez. O texto da tabela 6.2.1 foi submetido ao Sistema Robótico.

Tabela 6.2.1 – Instruções imperativas submetidas ao Sistema Robótico.

Eu quero que dê meia-volta, siga em frente por 1,2 metros, vire para a direita, avance devagar por 40 cm e depois fique 20 cm mais para a esquerda.

A tabela 6.2.2 apresenta a resposta do sistema ao texto da tabela 6.2.1. Todas as ações contidas na instrução imperativa foram corretamente interpretadas e foram completadas nos parâmetros faltantes conforme a mensagem de retorno. O passo nº 05 indica a resposta possível para o robô Curumim perante uma instrução de deslocamento para os lados.

Tabela 6.2.2 – Comandos e mensagens resultantes do texto da tabela 6.2.1.

Passo	Comando	Mensagens
01	Rotate(180,0)	OK
02	Move(1200,200)	Faltou informar a velocidade. Usando 200 como padrão
03	Rotate(-90,0)	OK
04	Move(400,100)	OK
05	Rotate(90,0)	Rotação implícita na instrução.
05	Move(200,200)	Faltou informar a velocidade. Usando 200 como padrão.

Outro teste de instruções na forma imperativa é o texto da tabela 6.2.3, também submetido ao Sistema Robótico. Considerar que a palavra “começo” é um local

de referência conhecido na planta, sendo onde começou a instrução da tabela 6.2.1. Também, a oração que contém esta palavra apresenta um problema para o sistema, que precisará encontrar o caminho para atingir o local referido.

Tabela 6.2.3 – Outras instruções imperativas submetidas ao Sistema Robótico.

Agora retorne para o começo, depois olhe para trás. Então, siga lentamente por mais 60 cm para esquerda e depois corra 1 metro para a direita.

A resposta do sistema ao texto da tabela 6.2.3 é mostrada na tabela 6.2.4 apresenta. A nova instrução imperativa foi corretamente interpretada e os parâmetros faltantes foram devidamente assumidos pelos valores padronizados para permitir a execução da melhor forma possível.

Tabela 6.2.4 – Comandos e mensagens resultantes do texto da tabela 6.2.3.

Passo	Comando	Mensagens
01	Rotate(-90,0)	OK.
02	Move(200,200)	OK.
03	Rotate(-90,0)	OK.
04	Move(1000,200)	OK.
05	Rotate(90,0)	OK.
05	Move(200,200)	OK.
06	Rotate(180,0)	OK.
07	SEM COMANDO	Nenhuma ação foi identificada.
08	Rotate(90,0)	Rotação implícita na instrução.
09	Move(600,100)	OK.
10	Rotate(-90,0)	Rotação implícita na instrução.
11	Move(1000,300)	OK.

Os passos de 01 a 05 referem-se ao caminho encontrado pelo sistema, que pode ser dito correto, considerando os obstáculos, para atingir o ponto de referência mencionado. A figura 6.2.1 mostra a Interface em Linguagem Natural logo após o processamento da instrução, antes da execução, onde é possível observar a planta do local e a rota traçada. Deve-se observar que o caminho traçado é somente aquele

encontrado pelo algoritmo de *Busca de Caminho* e não a totalidade do percurso a ser percorrido conforme consta da instrução do usuário, na tabela 6.2.3.

O trecho “Então” é considerado pelo sistema como uma oração e, por não conter nenhum significado aproveitável, surge como o passo 07. Os passos 08 e 10 são a resposta apropriada considerando o robô utilizado nos testes.

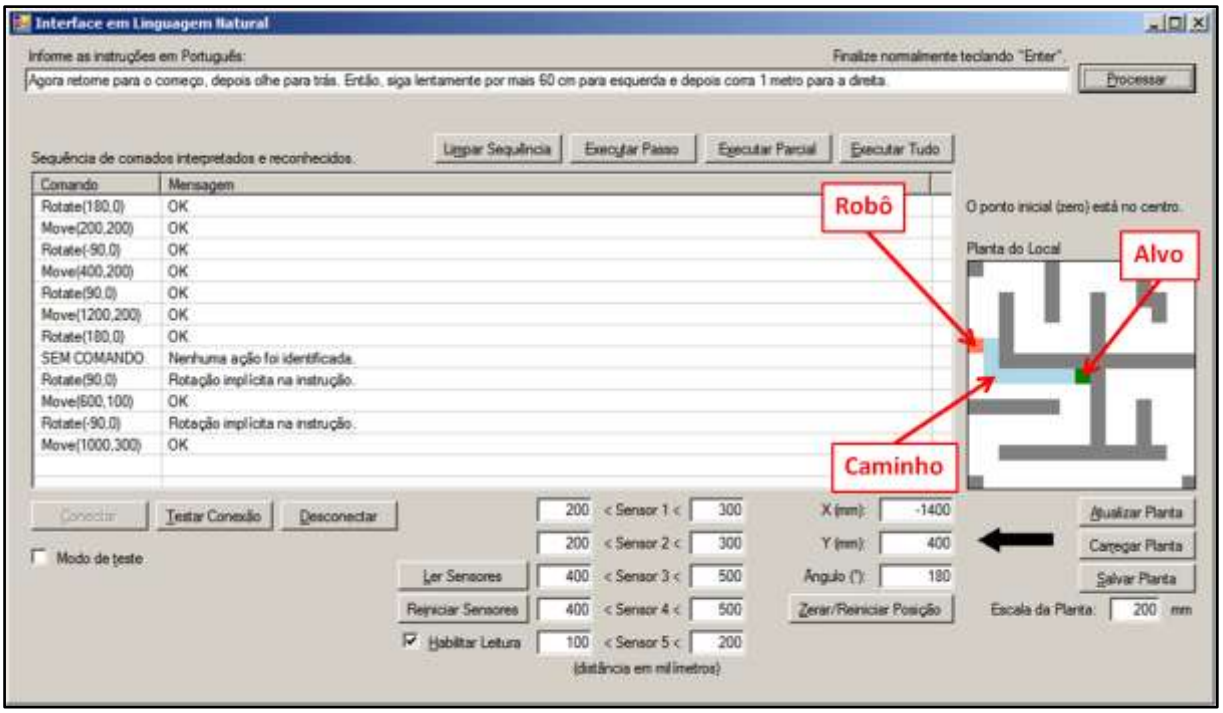


Figura 6.2.1 – Instrução imperativa e um caminho encontrado.

Para testar instruções na forma de outros tipos de afirmações, o texto da tabela 6.2.5 foi submetido ao Sistema Robótico. Nele, as ações de movimentação foram explicitadas.

Tabela 6.2.5 – Instruções afirmativas submetidas ao Sistema Robótico.

O manual recomenda que o robô desse meia-volta, que em seguida se movesse para frente 40 cm, depois corresse mais 80 cm e parasse 20 cm mais para a esquerda.

A tabela 6.2.6 apresenta a resposta do sistema ao texto da tabela 6.2.5, sendo que todos os comandos referem-se a ações descritas na instrução, somente completadas

pelos parâmetros faltantes conforme o caso. A rotação apropriada ao Curumim também foi adicionada corretamente, no passo 04. Não houve interferência de outros trechos das orações (como por exemplo: “O manual recomenda que…”) na interpretação daquilo que o sistema é capaz de executar.

Tabela 6.2.6 – Comandos e mensagens resultantes do texto da tabela 6.2.5.

Passo	Comando	Mensagens
01	Rotate(180,0)	OK
02	Move(400,200)	Faltou informar a velocidade. Usando 200 como padrão
03	Move(800,300)	Faltou indicar o sentido (frente ou trás). Usando frente com padrão.
04	Rotate(90,0)	Rotação implícita na instrução.
05	Move(200,200)	Faltou informar a velocidade. Usando 200 como padrão.

Sabendo que a palavra “início” é um local de referência conhecido pelo Sistema Robótico, coincidente com o ponto inicial da instrução da tabela 6.2.5, outro teste com instruções na forma de afirmação é o texto da tabela 6.2.7. Nesta instrução, o sistema deve primeiro resolver o problema do caminho, sendo que o usuário expressa o que precisa simplesmente como faria para outro humano.

Tabela 6.2.7 – Outra instrução afirmativa submetida ao Sistema Robótico.

Era para o robô ficar no ponto de início.

A tabela 6.2.8 apresenta a resposta do sistema à instrução da tabela 6.2.7. O sistema interpretou a instrução e resolveu o problema do caminho satisfatoriamente. A figura 6.2.2 exibe a Interface em Linguagem Natural após o processamento, antes da execução, onde é possível observar a planta e a rota.

Tabela 6.2.8 – Comandos e mensagens resultantes do texto da tabela 6.2.7.

Passo	Comando	Mensagens
01	Rotate(180,0)	OK.
02	Move(200,200)	OK.
03	Rotate(-90,0)	OK.
04	Move(1200,200)	OK.

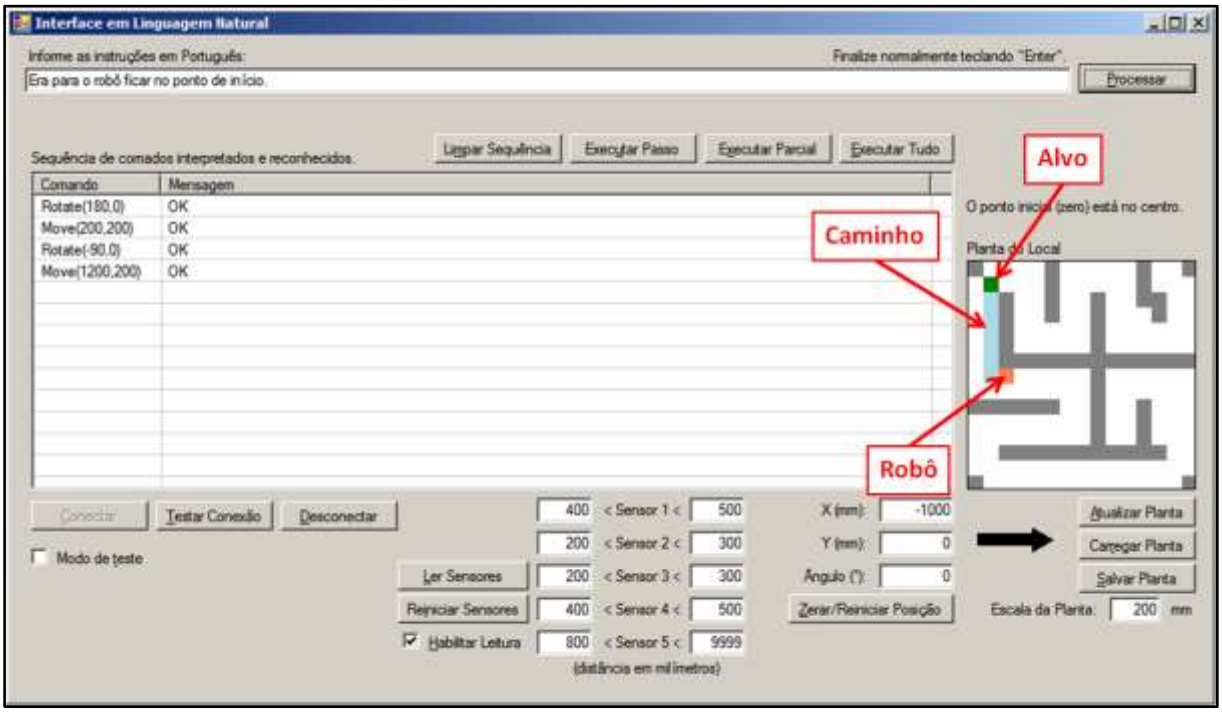


Figura 6.2.2 – Instrução afirmativa e um caminho encontrado.

Para verificar a capacidade de interpretação do sistema, todas as sentenças da tabela 6.2.9 foram submetidas e resultaram interpretadas corretamente, cada uma, como o mesmo comando pelo Sistema Robótico, indicado na tabela 6.2.10.

Tabela 6.2.9 – Várias maneiras de escrever a mesma instrução.

Frase	Texto
01	Robô: corra para frente 40 cm.
02	O robô precisa correr para frente mais 40 cm.
03	Correr em frente 40 cm é o que precisa fazer.
04	Para frente é para onde deve correr 40 cm.
05	Quero ver o robô correndo para frente à distância de 40 cm.
06	O chefe gostaria que o robô corresse para frente 40 cm.
07	Que para frente 40 cm corresse o robô é o que gostara o chefe.

Tabela 6.2.10 – Comandos e mensagens resultantes do texto da tabela 6.2.9.

Passo	Comando	Mensagens
01	Move(400,300)	OK.

6.2.2 TESTE 2 – REQUISIÇÕES NA FORMA INTERROGATIVA

Outra forma de comunicação entre Seres Humanos em Linguagem Natural são as construções na forma interrogativa. Em alguns casos, podem ser usadas como forma gentil de instruir um comando, com a ideia que sugerem. Como exemplo, escolhendo e reescrevendo algumas frases da seção 6.2.1, são sentenças do tipo:

- "Quer entrar na sala, por favor?";
- "Não é melhor você ir para a saída?"; e
- "Talvez não fosse melhor ficar mais para a esquerda?"

Em outros casos, expressam uma necessidade dos envolvidos, nos exemplos do item 6.2.1 reescritos, seriam frases como:

- "O manual não diz que este objeto deveria ficar na entrada?";
- "Quer vir mais para perto da janela para ver o evento?"; e
- "Avisaram que o chefe pediu para que você fosse à garagem?".

As questões também possuem várias camadas de significado em relação ao contexto humano. É possível dizer que na camada de significado de significado de ação de movimento, igualmente expressam o deslocamento de algo ou alguém de um lugar para outro como forma de resposta proativa.

Serão apresentados nesta seção os testes realizados com instruções construídas como interrogações e os resultados apresentados pelo Sistema Robótico proposto, pela sequência de comandos.

O texto da tabela 6.2.11 foi submetido ao Sistema Robótico, testando uma interrogação.

Tabela 6.2.11 – Instrução na forma interrogativa submetida ao Sistema Robótico.

O robô pode ir para o início agora?

Observar que o conteúdo profundo (significado) da interrogação da tabela 6.2.11 é o mesmo expresso pela instrução da tabela 6.2.7. Da mesma forma, a palavra início é uma referência conhecida pelo Sistema Robótico e coincidente com o ponto

inicial da instrução da tabela 6.2.5. A tabela 6.2.12 exibe a resposta do sistema ao texto da tabela 6.2.11, que é igual ao conteúdo da tabela 6.2.8.

Tabela 6.2.12 – Comandos e mensagens resultantes do texto da tabela 6.2.11.

Passo	Comando	Mensagens
01	Rotate(180,0)	OK.
02	Move(200,200)	OK.
03	Rotate(-90,0)	OK.
04	Move(1200,200)	OK.

Para testar uma interrogação com ação explícita, a instrução da tabela 6.2.13 foi submetida ao Sistema Robótico.

Tabela 6.2.13 – Outra instrução interrogativa submetidas ao Sistema Robótico.

Que tal ir 40 cm mais para trás?

A tabela 6.2.14 exibe a resposta do sistema ao texto da tabela 6.2.13. O único comando interpretado está correto, inclusive com o deslocamento negativo (para trás), já que o robô é capaz de executá-lo desta forma.

Tabela 6.2.14 – Comandos e mensagens resultantes do texto da tabela 6.2.13.

Passo	Comando	Mensagens
01	Move(-400,200)	Faltou informar a velocidade. Usando 200 como padrão.

6.3 DIFICULDADES ENCONTRADAS

A Linguagem Natural é por si só muito rica e flexível, sendo que uma mesma ideia pode ser expressa de inúmeras maneiras. Uma dificuldade encontrada foi o acesso a pessoas lusófonas com conteúdo culturalmente diferente para ser capaz captar uma variedade maior e estilos de expressão. Isto permitiria um planejamento prévio mais abrangente do léxico. Mais abrangentes também poderiam ser as avaliações do sistema, dessa forma.

Os modelos em língua portuguesa e os *Corpus* disponíveis gratuitamente usados no *SharpNLP*, tanto quanto os do *OpenNLP*, ainda são insuficientes para obtenção de resultados satisfatórios. A consequência é que a divisão de sentenças, a divisão de palavras/*tokens* e a marcação sintática resultam incorretas em maior proporção nos casos testados, em taxa bem inferior ao limite teórico. Isto leva o Sistema Robótico a interpretações incorretas ou à impossibilidade de interpretação do texto.

Por exemplo, a frase 01 da tabela 6.3.1 não pôde ser interpretada porque o Sistema Robótico não encontrou um local de destino, já que a palavra "entrada" foi marcada incorretamente como "verbo conjugado no particípio" (a marcação correta seria "substantivo"), e a frase 02 foi interpretada corretamente, já que sua redação não provocou o aparecimento do problema.

Tabela 6.3.1 – Frases processadas diferentemente pelos modelos em Português.

Frase	Texto
01	O chefe queria que você ficasse perto da entrada.
02	O chefe queria que você ficasse perto da saída.

Outro exemplo, exceto pelas frases de teste 04 e 05 da tabela 6.2.9, reescrevendo as demais com as expressões "para frente" ou "em frente" colocadas depois de "40 cm" ("40 cm para frente" ou "40 cm em frente"), o sistema reportava aviso de que a unidade de medida da distância não havia sido informada, assumindo um valor padronizado.

Analisando o problema, encontrou-se a incorreta marcação de "cm" como "Verbo no infinitivo". O mesmo aconteceu ao tentar usar "m" (metro), porém não ao usar "mm". Substituindo as abreviações "cm" e "m" pelas respectivas palavras "centímetros" e "metros", as marcações sintáticas voltaram a ser corretas. Convém observar que as frases 03 e 07 também resultaram em marcações corretas, mesmo com "cm" sendo usado no meio das demais palavras ou *tokens*.

Também, ao tentar colocar o ponto de abreviatura, observou-se a separação incorreta da sentença, impossibilitando a interpretação. Com isto, conclui-se que essas abreviaturas não se encontram presentes com a devida frequência, tanto no *Corpus* de treinamento para marcação sintática quanto no de treinamento para separação de sentenças.

Como sugestão para desenvolvimento futuro, a solução é a elaboração de um *Corpus* de treinamento e um *Corpus* de teste suficiente para treinamento dos Modelos de Entropia Máxima do *SharpNLP*. Isto requer, pelo menos, a verificação manual de quantidade enorme de frases previamente marcadas, em volume superior a alguns milhões de palavras.

Também decorrente dos modelos em Português disponíveis, as redações com o sujeito da oração oculto ocasionam mais erros na marcação dos verbos escritos no imperativo, já que esta forma verbal surge facilmente nas redações de instrução. Uma vez que o *Corpus* usado para treinar os modelos disponíveis gratuitamente é baseado em texto predominantemente jornalístico, onde os tempos verbais mais comumente encontrados são as terceiras pessoas, do singular e do plural, no passado e no presente, o imperativo é raro. Como a conjugação na língua portuguesa dá origem a palavras diferentes, elas não aparecem com frequência suficiente no *Corpus* para permitir a máxima entropia necessária para os modelos, ou simplesmente não aparecem, e resultam classificadas com marcação sintática incorreta.

6.4 LIMITAÇÕES

6.4.1 VERBOS COMO BASE DA TEORIA DOS ESQUEMAS DE CASOS

Na elaboração das instruções para o Sistema Robótico, é extremamente recomendável que os verbos de ação sejam explícitos, ou seja, estejam presentes e não ocultos, uma vez que são os primeiros constituintes da frase a serem analisados. Por exemplo, redações como os exemplos abaixo acarretam interpretação incorreta:

- Seguir para frente e para a direita.
- Sobre a movimentação, para a entrada e depois 40 cm para a esquerda.

6.4.2 MUDANÇA NA ORDEM DOS COMANDOS

Os recursos para encadeamento temporal dos comandos na instrução são limitados à sequência em que foram escritos. Ou seja, expressões que indicam mudança de antecedência entre eventos, como “antes de” e “depois de” não são interpretadas nem tampouco afetam a ordem de execução. Por exemplo, nas instruções abaixo, a execução correta seria primeiro “virar a direita” e depois “seguir em frente”, porém o Sistema Robótico executará o contrário:

- Antes de ir em frente, vire à direita.
- Siga em frente depois de virar à direita.

Como sugestão de desenvolvimento futuro, seria desenvolver o trecho de software capaz de executar o encadeamento temporal correto, ou seja, a ordem dos comandos de acordo com a redação.

6.4.3 SOBRE AS QUANTIDADES NUMÉRICAS

Para garantir que os algoritmos estocásticos do *SharpNLP* classifiquem gramaticalmente as quantidades numéricas de forma correta, elas precisam ser escritas por meio de números propriamente ditos e não por meio de palavras (como "sessenta", por exemplo). Uma sugestão de desenvolvimento futuro seria treinar os Modelos de Entropia Máxima com *Corpus* preparado para melhores resultados na classificação de números e também dotar o Sistema Robótico, e o léxico, de recursos para interpretar o significado das palavras (quantidades numéricas).

6.4.4 NOVOS LOCAIS DE REFERÊNCIAS E MEMÓRIA DE CONTEXTO

O sistema possui como informações somente as instruções fornecidas pelo usuário, a planta do local onde o robô é testado e os pontos de referência internos. Não possui memória de contexto ou recursos de aprendizagem através da Interface em Linguagem Natural. Desta forma, não é possível definir uma referência nova com uma instrução do tipo "A recarga fica no quadrado 4,10. Vá para a recarga.". Uma sugestão de desenvolvimento futuro, seria a implementação do reconhecimento de instruções que definem novas referências (nomes de lugares) e formas de gerenciar memória de contexto de trabalho ou operação.

6.4.5 USO DE COORDENADAS

Como a presente dissertação tem por objetivo estudar o processamento da linguagem natural conforme ela é utilizada por pessoas leigas, o Sistema Robótico não foi dotado da capacidade de interpretar instruções de movimentação por coordenadas. Assim, instruções do tipo "Você deve ir até o ponto 13,7" não serão interpretadas corretamente, mas será interpretada da mesma forma que uma instrução elementar do tipo "movimentar" (prescindindo dos parâmetros de distância, sentido ou velocidade). Uma sugestão de desenvolvimento futuro seria incluir a capacidade de reconhecer instruções de movimentação por coordenadas.

6.4.6 USO DE CORREFERÊNCIAS

Um problema linguístico cuja solução não foi implementada no Sistema Robótico proposto são as correferências. Assim, o trecho "ao mesmo lugar" (referindo-se à janela) do exemplo a seguir não poderá ser interpretadas corretamente:

- "O robô deve ir até a janela, virar à direita e andar 1 metro. Depois deve retornar ao mesmo lugar, dar meia-volta e..."

Uma sugestão de desenvolvimento futuro é dotar o sistema de recurso para resolver correferências.

6.4.7 BUSCA DE CAMINHO À PARTIR DE PONTO FUTURO

Para correto funcionamento do recurso de Busca de Caminho, é importante observar que seja a primeira tarefa da instrução e que também seja colocada somente uma única por texto do usuário. O algoritmo de Busca de Caminho foi implementado tomando como início o local atual do robô. Além disso, o sistema não possui memória ou processamento de contexto mais elaborados. Assim, uma busca de caminho a partir de um ponto futuro exigiria recursos de planejamento mais elaborados, que podem ser agregados ao sistema, sendo outra sugestão de desenvolvimento futuro.

6.4.8 USO DE INSTRUÇÕES NEGATIVAS

A interpretação de afirmativas pode ser executada com maior sucesso e também a interpretação de algumas interrogativas. Porém, interessantes questões surgem quando se analisa a interpretação de negativas. Em grande parte dos casos, as negativas ocasionam várias ambiguidades, incorrendo na necessidade de outras informações complementares para permitir a correta interpretação e posterior execução. Por exemplo, a negativa abaixo:

- "Não vá para frente."

O que realmente significa? Significa que se deve ir para trás ou é preciso ficar parado? E ficar parado até quando? É permitido ir para os lados? É possível dar meia volta e ir no mesmo sentido de ré? Depois do primeiro movimento permitido será possível ir para frente? Ou então, a partir de quando será possível ir para frente outra vez? Assim, uma instrução negativa, que pode significar algum tipo de regra de restrição, pode ser abordada como sendo um problema ou enigma a ser solucionado por recursos

mais elaborados de inteligência. De maneira geral, é possível considerar que o uso de instrução negativa sugere a transferência de problema (ou problemas) e não a indicação clara de algo que precisa ser feito. Ao lado disso, vale considerar que a presença da palavra "não" nem sempre indica uma negativa e pode consistir em mero recurso estilístico de linguagem, como nos exemplos a seguir:

- "Não é por menos que o manual instruiu o robô a seguir adiante mais 1 metro!"
- "É melhor ficar 10 cm mais para esquerda, não?"

Também decorrente da complexidade envolvida na interpretação de negativas, a negação do texto abaixo será ignorada, levando o robô para uma posição final bem diferente da pretendida:

- "Não quer ver o robô indo para a esquerda, mas indo para a direita."

Como o Sistema Robótico ignora as negativas, o texto será interpretado erroneamente como sendo composição dois comandos: o primeiro para a esquerda e o segundo para a direita. Ou seja, o robô executará uma rotação para a esquerda, avançará uma distância padrão na velocidade normal, depois fará rotação para a direita e avançará uma distância padrão na velocidade normal. Ele terminará num ponto 45 graus à esquerda e à frente do ponto inicial (se não houver obstáculos), enquanto na instrução do usuário deveria terminar a uma distância padrão à direita.

6.4.9 USO COMBINADO DE COMANDOS

No encadeamento de comandos, para serem corretamente identificados na instrução, eles devem ser indicados pelo menos pelo verbo, um a um, sendo que os parâmetros podem ser omitidos caso o valor padrão for aceitável. Instruções como o exemplo abaixo não serão interpretadas corretamente, pois o segundo comando foi escrito sem o verbo:

- Seguir para frente e para a direita.

É oportuno observar que dependendo da quantidade e do tipo dos parâmetros de um comando, ao serem detalhados na instrução do usuário há alguma possibilidade de o sistema identificar o comando pela verificação contextual. Porém, à

medida que novos comandos forem agregados, a probabilidade de a interpretação ser ambígua aumentará e a possibilidade de identificação diminuirá. Por exemplo, ao reescrever o exemplo anterior detalhando mais parâmetros, o Sistema Robótico passa a ser capaz de interpretar o segundo comando:

- Seguir para frente e para a direita mais 40 cm rápido.

Outra maneira comum de combinar comandos na linguagem natural é com o objetivo de criar restrição ou fazer filtragem. São instruções como o exemplo abaixo:

- Caminhar até a porta que fica à direita.

Instruções como esta não podem ser interpretadas corretamente. Por se tratar de um problema e, para que seja devidamente resolvido, uma possível solução seria: saber o que são portas e como reconhecê-las; saber quais as portas que estão à direita; saber qual é a mais próxima; e, por fim, buscar o caminho até ela. Requer, portanto, outro tipo de processamento que não uso simples dos comandos Rotear e Rotacionar.

Uma maneira de combinar comandos, mesmo que a combinação não seja muito aparente, são instruções do tipo:

- Vá para cada uma das portas.

A interpretação do Sistema Robótico será incorreta. Nesta instrução, é pouco provável que um único lugar de referência seja chamado de “portas”. Caso exista, e precisará necessariamente ser único, o robô será direcionado para aquele local especificamente e o comando concluído.

Uma instrução assim se constitui em um problema a ser resolvido ou um serviço a ser executado. Para que seja passível de execução pelo sistema, uma possível solução seria: saber o que são portas e como reconhecê-las; saber quais as portas que existem na planta; colocá-las em ordem e estabelecer um itinerário de visitação; buscar o caminho até elas, uma a uma; e, por fim, executar a visitação. Requer, portanto, outro tipo de processamento que não uso simples do comando Rotear.

6.4.10 NOME DAS REFERÊNCIAS

Caso existam locais de referências com mesmo nome (como mais de uma "saída", por exemplo), eles precisam ser identificados com nomes únicos para se tornarem referências claras, sem ambiguidade.

Também, os locais de referência precisam ser identificados por uma única palavra ou então palavras combinadas por hífen (por exemplo: "guarda-louça", "porta-A", "saída-2", "porta-principal" etc.). Não são permitidos nomes com duas ou mais palavras separadas por espaços.

6.5 OBSERVAÇÕES FINAIS

Durante a realização dos testes, uma funcionalidade que surgiu sem ser intencional durante o planejamento foi a capacidade de responder perguntas. No caso, trata-se de uma funcionalidade bastante restrita e pouco abrangente, mas que é digna de ser mencionada tendo-se em vista que o Sistema Robótico possui várias informações, elas podem ser utilizadas em conjunto e o usuário naturalmente terá necessidade de conhecer ou pesquisar esse conteúdo.

É interessante notar que perguntas iniciadas com "Onde é/está/fica" e "Como chegar a" uma dada referência conhecida, o sistema é capaz de apresentar resposta satisfatória, ainda que pelos meios disponíveis na Interface em Linguagem Natural e não a linguagem natural propriamente dita.

Assim, ao colocar a primeira pergunta da tabela 6.5.1 para o Sistema Robótico por meio da Interface em Linguagem Natural, após o processamento, antes da execução, a resposta foi conforme mostra a figura 6.5.1. A segunda pergunta apresenta o resultado como exibido na figura 6.5.2.

Tabela 6.5.1 – Perguntas submetidas ao Sistema Robótico em teste.

Pergunta	Texto
01	Onde fica a entrada?
02	Como chegar até a saída?

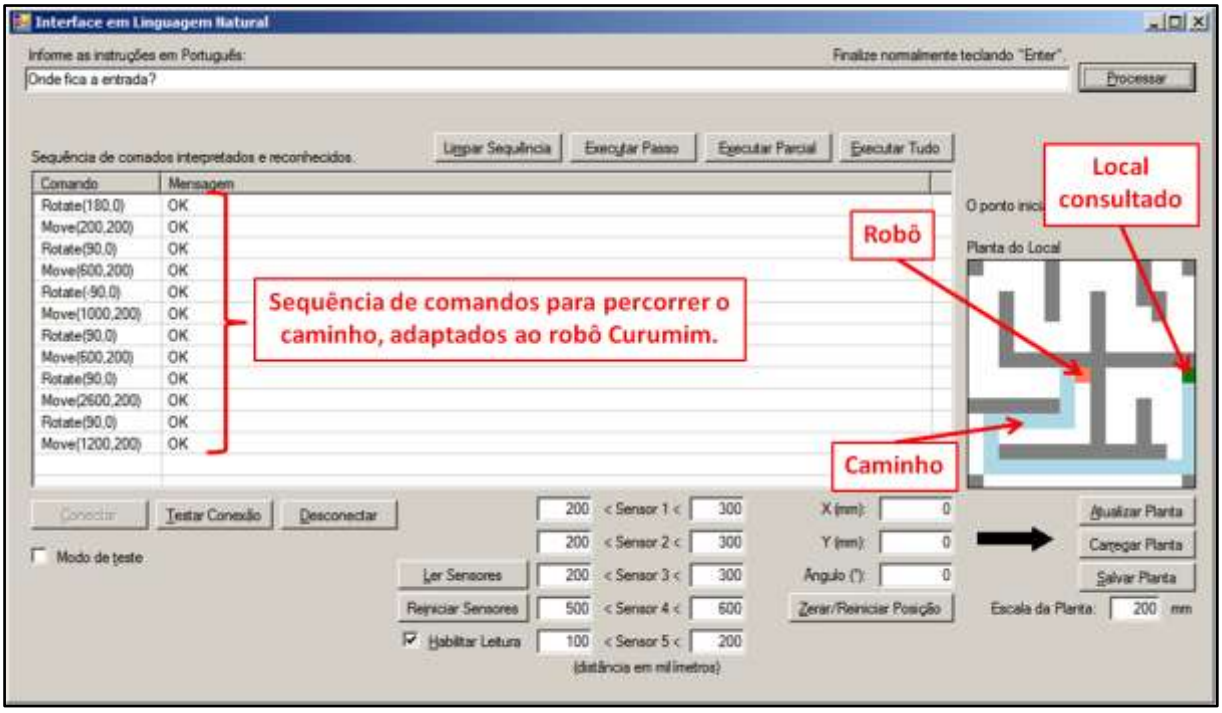


Figura 6.5.1 – Teste de resposta para perguntas do tipo "Onde é/está/fica ...".

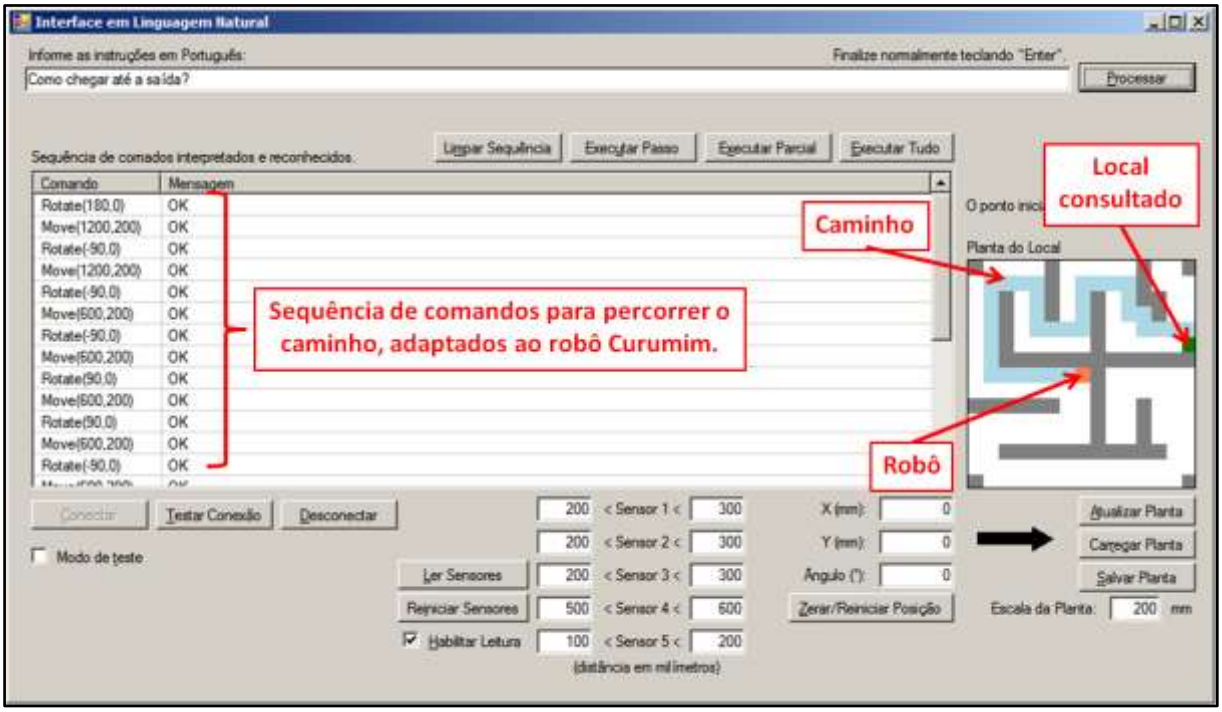


Figura 6.5.2 – Teste de resposta para perguntas do tipo "Como chegar até...".

Do ponto de vista da interação entre as partes envolvidas, ou seja, o usuário e o sistema, a resposta é razoavelmente natural e faz sentido ao usuário humano desde que o ponto de referência é o lugar do próprio robô e a indicação do caminho é apresentada de duas formas diferentes, sendo a gráfica a mais conveniente para humanos.

Comparativamente, se esta pergunta fosse feita a outro humano, e não ao sistema robótico, a resposta inicial, se conhecida, provavelmente seria menos específica sobre caminho e seriam mencionados pontos de referência próximos, do destino, do caminho ou de ambos, simplificando o volume de informações a transmitir. Também é bastante comum no processo de comunicação humano a interação entre os interlocutores, principalmente quando o caminho a ser indicado é complexo como o da figura 6.5.2. A experiência geralmente mostra às pessoas que é eficiente transmitir respostas complexas em partes menores, avaliar a compreensão de cada parte, adaptar ou corrigir sempre que for identificado algo errado e constantemente avaliando o conhecimento das referências que o destinatário da mensagem possui. Um humano nem sempre dispõe de recursos para responder por meio de um mapa do caminho sobre a planta, enquanto o Sistema Robótico dispõe da Interface em Linguagem Natural mas não de recursos para elaborar respostas em linguagem natural.

O sistema robótico, neste caso, comportou-se como se fosse uma única organização fisicamente contida no local do robô Curumim. Porém, como a comunicação entre o computador hospedeiro do *software* do Sistema Robótico e o Curumim é feita por rádio, eles estão fisicamente separados e, mais propriamente, o sistema está margeando a ubiquidade ao romper os limites da presença física constrita a um único ponto. Se fosse utilizado mais de um robô, haveria o problema de ambiguidade de referência e a resposta seria diferente, dependendo da estratégia implementada internamente para resolver esta ambiguidade.

A capacidade de responder estes tipos de perguntas se deve à estratégia de executar a interpretação do texto o mais proativo possível, utilizando um mínimo de recursos linguísticos. Ou seja, partindo de algumas palavras comuns entre o léxico do sistema e as escolhidas pelo usuário para redação do texto, é feita uma avaliação do Esquema de Caso mais provável associado a um comando ou serviço de que dispõe. Em seguida, é simulada a execução deste comando ou serviço e o resultado é submetido à avaliação do usuário. A interpretação da pergunta como tal pelo sistema não acontece de

fato, ou seja, não há nada no sistema que o faça compreender o significado das palavras “Onde fica” ou “Como chegar” da forma como significam para humanos lusófonos. O comportamento do sistema tem significado como resposta graças à abstração que ocorre do ponto de vista do usuário com esta condição de uso. Esta abstração poderia ser mais estudada e melhor aproveitada para criar outros serviços inteligentes e úteis, em sugestão de trabalhos futuros.

7. DISCUSSÃO, CONCLUSÃO E TRABALHOS FUTUROS

7.1 DISCUSSÃO

7.1.1 CONTEXTUALIZAÇÃO

Importante extensão da capacidade criativa e transformadora do Ser Humano principalmente após a Revolução Industrial, as máquinas se tornam cada vez mais companheiras imprescindíveis no cotidiano. Há bastante tempo são usadas em atividades como extrair, transportar, separar, unir, perfurar, aquecer, refrigerar, cozinhar entre outras. Mais recentemente integraram-se à vida comum auxiliando e participando de atividades como estudar, ensinar, aprender, trabalhar, guardar, organizar, recuperar, esconder, pesquisar, comunicar, pagar e muitas outras mais. Inicialmente, mais assemelhadas a ferramentas mais versáteis, auxiliavam nas transformações físicas, puderam ser dotadas da capacidade de representar e processar conteúdos abstratos por meio do *software*. À medida que se tornam mais amigáveis e convenientes às atividades humanas, as máquinas são integradas e agregam considerável valor em termos de produtividade e economia de tempo e esforço.

Quanto mais natural e intuitivo for para os Seres Humanos a forma de interação, maior será a participação delas. E máquinas capazes de participar das atividades tanto físicas quanto abstratas tornar-se-ão cada vez mais presentes.

7.1.2 REALIZAÇÃO DO SISTEMA ROBÓTICO PROPOSTO

A concepção, desenvolvimento e testes de um sistema de supervisão, operação e controle de robôs utilizando linguagem natural somente foi possível em decorrência do estudo, adaptação e aplicação de teorias e tecnologias existentes. Ele foi composto de um *robô móvel* nacional (o Curumim) e um *software* de integração. Este último foi elaborado tanto com funções genéricas previamente desenvolvidas e disponibilizadas pela comunidade de *software* livre, quanto com partes desenvolvidas especialmente para desempenhar todas as demais funcionalidades necessárias.

Os testes mostram que é possível o uso da linguagem natural para acionamento de robôs por usuários leigos e não treinados tecnicamente em robótica.

O uso da teoria dos Esquemas de Casos ("*Case Frames*") de *Fillmore* mostrou-se conveniente na interpretação do significado do texto em linguagem natural para mapeá-lo às funcionalidades determinadas para o sistema em função do robô sendo utilizado.

O uso do processamento estatístico da linguagem natural por meio de *Modelos de Entropia Máxima* apresentou razoável dificuldade e resultados ainda imprecisos. Há dificuldade de acesso a *Corpus* para treinamento a fim de que seja realizada separação de sentenças, de palavras ou *tokens* e para marcação gramatical das palavras com taxas de erro pequenas e próximas aos limites teóricos. Os disponíveis gratuitamente são insuficientes para uma aplicação mais satisfatória. A elaboração de um *Corpus* específico para este fim, por outro lado, requer enorme quantidade de esforço, não sendo possível sua realização sem um trabalho cooperativo e coordenado.

Desta forma, cabe a recomendação para o aprofundamento do estudo da teoria dos Esquemas de Casos sem a prévia marcação gramatical das palavras, ou então avaliar a utilidade de outras formas de marcação gramatical das palavras e *tokens*.

O uso de Gramática Livre Contexto e da Gramática Semântica também sugere ser interessante complemento na extração de dados e informações das orações, para preencher os papéis (parâmetros) dos esquemas de casos.

7.1.3 INTEROPERABILIDADE

O sistema pode ser adaptado para incluir outros robôs, sendo preciso agregar novas partes de *software* adequadas ao seu controle e comando, conforme suas características funcionais.

Cada nova funcionalidade agregada ao sistema proposto poderá interagir com as já existentes. Assim, um ciclo de análise e planejamento de novas funcionalidades deverá ser executado, tendo-se em vista a potencialidade desta interação, principalmente com as novas que podem surgir por meio da abstração e da atribuição de significado a estados ou dados. Exemplo disto foi o uso de algoritmo de Busca de Caminho.

Foi depreendido esforço em tornar o sistema flexível, no sentido de que fosse possível adaptá-lo facilmente às necessidades, novas funcionalidades e novos robôs. Foi conseguido sucesso em propor um algoritmo de *Identificação de Comandos e Parâmetros*, que executa uma *Avaliação de Contexto das Orações* e posteriormente uma *Avaliação de Significado e Extração de Informações*, utilizando um léxico armazenado em

arquivo. Isto torna a adaptação do vocabulário e ajuste do reconhecimento de comandos a partir da interpretação do texto em linguagem natural mais conveniente, não sendo necessário alterar o código fonte do programa.

Contudo, modificações nas funcionalidades ou inclusão de outras novas no sistema poderão ser executadas somente por usuário tecnicamente habilitado, capaz de compreender a linguagem de programação, o funcionamento do robô e o funcionamento do sistema em si.

O uso da linguagem natural por usuários humanos não é só para transmitir instruções diretas do que se deseja ver executado, mas frequentemente transmite problemas a serem resolvidos para determinar o que realmente precisa ser executado depois. Não basta somente maior vocabulário, com melhoria do léxico, para tornar o sistema mais abrangente. A capacidade de interpretação do significado do texto e a capacidade de execução de novas ações e serviços, ou a de solucionar problemas, depende bastante do que está disponível em termos de *software*, como formas de abordagem para ambiguidades, uso de convenções e falta de detalhamento no texto, bem como também depende de recursos heurísticos, como mapas, tabelas, valores de referência e valores padronizados.

7.1.4 FOCO EM SERVIÇOS

As soluções encontradas com as teorias e tecnologias aplicadas no desenvolvimento do Sistema Robótico proposto mostraram-se satisfatórias na capacidade de compor formas de reconhecer e executar serviços colocados à disposição do usuário através da Interface em Linguagem Natural.

O sistema é capaz de interpretar e executar corretamente instruções afirmativas em linguagem natural como “Quero que o robô vá para perto da mesa.”, “O robô deve ir até a entrada e ficar 20 cm mais para a esquerda.”, interrogativas como “Não seria melhor se ele fosse para a saída?” e mesmo capaz de apresentar uma reação útil, como se fosse uma resposta, para pergunta do tipo “Onde está o abajur?”.

Caso tivesse sido implementado no *software* de integração o recurso de captura de imagem e vídeo disponível no robô Curumim, seria possível agregar novo serviço como reagir significativamente a instruções interrogativas do tipo “Como está a janela?”, quando já seria útil ir até o local e enviar uma imagem, por exemplo.

7.1.5 POSSIBILIDADES DE APLICAÇÃO DAS SOLUÇÕES ENCONTRADAS

As fontes teóricas e tecnológicas encontradas, bem como a soluções encontradas durante o desenvolvimento deste trabalho, apresentam-se promissoras para reutilização em outros trabalhos e aplicações futuras.

Não sendo linguagem tradicional para aplicações de Processamento da Linguagem Natural, a exploração das possibilidades do uso da linguagem de programação C# mostrou-se promissora. Há algumas iniciativas de *software* de código aberto e de *software* livre, como o próprio *SharpNLP*, que são úteis para pesquisas e desenvolvimentos de trabalhos na área. E outros mais virão no futuro. Os ambientes de desenvolvimento integrado para o C# estão disponíveis gratuitamente para trabalhos individuais e de pequena escala, como também se encontram ambientes de desenvolvimento integrado para escala profissional, como é o caso do *Visual Studio*®.

Até o momento da conclusão da redação deste trabalho, na pesquisa de fontes bibliográficas e de material de referência técnica, os trabalhos encontrados de aplicação prática envolvendo programação de computadores e a teoria dos Esquemas de Casos em língua Portuguesa direcionados ao processamento da linguagem natural para extração de significado não são muitos. Contudo, deve-se considerar que o acesso a este tipo de documentação técnica lusófona é consideravelmente restrito em relação ao acesso a documentos em língua inglesa, por exemplo.

7.2 CONCLUSÃO

O sistema proposto compõe-se de um robô nacional e um *software* de integração, dotado de recursos de interface, controle e inteligência. Este último é fruto exclusivo da realização deste trabalho.

Através da análise dos resultados obtidos durante os testes pode-se concluir que é possível conceber e realizar com sucesso um sistema de supervisão, operação e controle de robô baseado em linguagem natural por meio das teorias e tecnologias utilizadas, uma vez que as instruções escritas em Português foram satisfatoriamente interpretadas nos comandos do robô em cerca de 75% dos testes (solicitações na forma de instruções em Português) conforme pode ser observado pela média e mediana na tabela 6.1.5. Foram consideradas as variantes de redação permitidas pela língua (tais como: tempos verbais infinitivo, imperativo e subjuntivo; segunda e terceira pessoas;

afirmativas, interrogativas, forma direta e forma inversa) e desconsideradas as formas de expressão fora do escopo deste trabalho (que poderão ser incluídas em trabalhos futuros).

Também foi demonstrado com sucesso ser possível colocar um serviço útil ao dispor de usuários humanos: a navegação ou movimentação por referências externas convencionadas e nomeadas. Sua utilização significou a simplificação da comunicação necessária em linguagem natural, sendo que também tornou a redação da instrução mais próxima da forma habitual humana.

Mesmo assim, muito ainda precisa ser desenvolvido para aproximar a capacidade de interpretação do sistema robótico proposto à habilidade natural humana de comunicação.

7.3 CONTRIBUIÇÕES

Por ser tarefa complexa, que requer aplicação de apreciável diversidade de conhecimentos, a extração de significado de textos em linguagem natural pode ser abordada de melhor forma dividindo-se o grau de elaboração do texto natural em níveis de recursos possíveis de serem interpretados. A solução prática deste trabalho identificou e foi capaz de interpretar as formas naturais de expressão mais comumente utilizadas nas interações que requerem comunicação de instruções.

Considerando que este trabalho trata de uma *aplicação prática* envolvendo *programação de computadores utilizando C#* e a *teoria dos Esquemas de Casos* em *língua Portuguesa* direcionados ao *processamento da linguagem natural* para *extração de significado*, sua principal contribuição não está na proposição em si de um sistema para controle de robôs através de linguagem natural, mas sim na forma como executar o processamento da linguagem natural, que neste caso apresenta alguns caminhos para obtenção de resultados realmente úteis com uma língua falada por tantos no mundo. A solução prática utilizando C#, que não é linguagem de programação tradicionalmente utilizada na área de inteligência artificial, também agrega novas experiências e conquistas úteis aos pesquisadores e desenvolvedores interessados inclusive em aplicações profissionais.

O Sistema Robótico proposto apresenta-se como contribuição também nos recentes campos de aplicação da teleoperação e a telerrobótica, ao permitir forma

adicional de operar e controlar robôs, de maneira mais conveniente e útil, além do conceito tradicional de "imitação", empregado com mais frequência. Com recursos inteligentes, um usuário poderá informar "o que" deseja executar e o sistema será capaz de resolver o problema de "como" executar. A seção 5.5.3 (Planta do local e algoritmo de busca de rota) contém mais esclarecimentos. Para tanto, deve-se notar que inexiste restrição quanto à localização física do robô controlado por este sistema em relação ao lugar do seu usuário (ou, em possível condição futura: localização do**s** robô**s**).

O desenvolvimento de um sistema capaz de obter resultados práticos utilizando robô brasileiro contribui para as tecnologias nacionais tanto quanto contribui para evidenciar o nível de maturidade e desenvolvimento atingido pelo esforço de cientistas, profissionais e indústria do país.

As soluções propostas, as teorias e as tecnologias aqui abordadas permitem o desenvolvimento de formas mais naturais de usuários humanos operarem e controlarem robôs.

7.4 TRABALHOS FUTUROS

7.4.1 ESCLARECIMENTO

Para realização deste trabalho e solução dos problemas envolvidos, várias ideias e novas oportunidades de desenvolvimento ou melhoria surgiram. São agrupadas a seguir aquelas que foram naturalmente identificadas durante o desenvolvimento do trabalho como sugestão de desenvolvimento no futuro.

7.4.2 MULTIUSUÁRIO E MULTIRROBÔ

O Sistema Robótico, com o nível de desenvolvimento atingido ao longo deste trabalho, possibilita a utilização de uma única Interface em Linguagem Natural e um único robô. Porém, ele tem potencialidade para dispor de mais de um tipo de interface com os usuários, como interface via *Web*, telefone celular do tipo *Smartphone* ou telefone celular via SMS (mensagens curtas ou torpedos), por exemplo. O estudo dos recursos como *recebimento das solicitações*, *endereçamento das interfaces com os usuários e com os robôs*, *gerenciamento da execução das tarefas* e a possibilidade de *trabalho cooperativo entre os robôs* estão entre os assuntos interessantes que podem ser explorados.

7.4.3 UTILIZAÇÃO DE NÚMEROS NA FORMA LITERAL

Durante o desenvolvimento, deparou-se com limitação do treinamento dos Modelos de Entropia Máxima, que não permitem a correta classificação sintática forçando a utilização da grafia de quantidades numéricas somente com números. Uma melhoria útil consiste na adição de recursos para ser possível reconhecer e processas números também quando escritos na forma literal (como “um”, “dois”, “três”, “vinte”, “setenta” etc.). Não só partes do *software* precisam ser acrescidas, para transformar as palavras em números propriamente ditos, como o léxico precisará ser atualizado. Novo *Corpus* de treinamento precisará também ser elaborado, ou então a melhoria dos que estão disponíveis gratuitamente.

7.4.4 INSTRUÇÕES DE MUDANÇA NA ORDEM OU SEQUÊNCIA DOS COMANDOS

Na composição de instruções a serem executadas, podem surgir casos em que o usuário utilize expressões como “antes de” e “depois de”, que alteram a ordem de execução dos comandos. Incluir os recursos de *software* e de léxico necessários para interpretação e execução deste tipo de instrução aumentará a flexibilidade e utilidade do sistema.

7.4.5 DESLOCAMENTO POR MEIO DE COORDENADAS

Para a descrição do deslocamento do robô móvel é possível interpretar tanto instruções autorreferenciadas (por exemplo: “Ir para frente 10 cm.”) como as baseadas em referência previamente estabelecida e nomeada (por exemplo: “Ir para a porta.”).

Um recurso útil será dotar o sistema da capacidade de interpretar instruções baseadas em coordenada (por exemplo: “Ir para o ponto 4,7” ou “Girar para a direita 45 graus”). Neste caso, o léxico precisará ser atualizado tanto quanto o *software*.

7.4.6 MEMÓRIA DE CONTEXTO E RESOLUÇÃO DE CORREFERÊNCIAS

A língua portuguesa dispõe de recursos que simplificam a redação (e a comunicação) por meio de referências a conteúdos mencionados previamente, como as palavras e expressões “este”, “aquele” e “o mesmo”, entre outras formas de composição. Contudo, o Sistema Robótico não dispõe de meios para resolver tais referências. Tampouco, não é possível tirar proveito ou compartilhar informações contidas entre duas orações ou frases distintas.

Desta forma, estudos e desenvolvimentos de formas para gerenciar memória de contexto e para resolver correferências podem agregar funcionalidade e conveniência ao sistema.

7.4.7 APRENDIZADO PELA INTERFACE EM LINGUAGEM NATURAL

A Interface em Linguagem Natural com o usuário permite o controle e a supervisão do robô, mediante informações apresentadas para o usuário. É possível compor instruções que mencionam locais conhecidos e que foram armazenados previamente no léxico.

Uma possibilidade de melhoria útil seria a adição do reconhecimento e execução de instruções que permitam ao usuário adicionar e nomear novos pontos de referência através de linguagem natural.

Outra possibilidade de melhoria útil seria nomear os novos pontos de referência diretamente sobre a planta. Como o algoritmo de Busca de Rota (ou caminho) possui certa independência de funcionamento em relação aos de processamento da linguagem natural, esta integração precisará ser elaborada para que os resultados necessários sejam obtidos.

7.4.8 ALGORITMOS DE BUSCA HEURÍSTICA DE SOLUÇÃO

Os algoritmos de busca, entre os quais está o utilizado para Busca de Caminho (ou Rota) no sistema robótico, apresentam-se também em diversas variantes. É sugerido o estudo dessas variantes e sua aplicabilidade em casos específicos de utilização, não só na robótica, como também na linguística computacional.

Em particular, sugere-se o estudo e implementação do algoritmo de busca D* *Lite* (*D star lite*) no sistema proposto, por ser mais moderno e consumidor de menores recursos computacionais.

7.4.9 BUSCA DE ROTA OU CAMINHO A PARTIR DE UM PONTO FUTURO

Para superar as limitações atuais, a adaptação do sistema para permitir a busca de caminho a partir de um ponto futuro, ou mesmo um ponto qualquer, tornará o sistema para mais flexível frente às condições reais de uso que podem surgir.

Isto por que é necessário que a Busca de Rota seja a primeira instrução, ou seja, deve-se considerar sempre o local atual do robô para que o caminho encontrado esteja correto, e também não é possível utilizar mais de uma vez por instrução este

recurso de busca heurística, na forma como se apresenta neste último estágio de desenvolvimento.

7.4.10 ORAÇÕES COM SUJEITO OCULTO

Recurso linguístico possível no Português é a redação de frases com sujeito oculto. Ou seja, considerando-se que os componentes básicos de uma oração são *Sujeito*, *Verbo* e *Predicado*, ocultar o sujeito é reescrever a frase de exemplo "Eu comi uma fruta" como "Comi uma fruta", onde o pronome "Eu" foi suprimido na segunda redação. Não são todas as línguas que dispõem desta possibilidade.

No processamento estatístico da língua, o uso de sujeito oculto na oração ocasionou incorreções adicionais na classificação gramatical das palavras, especialmente os verbos.

Fica como sugestão o aprofundamento dos estudos para possível implementação de recursos ao Sistema Robótico proposto, para resolver as questões inerentes deste estilo de redação do ponto de vista da linguística computacional.

7.4.11 O DICIONÁRIO DE PALAVRAS RARAS DO SHARPNLP

Durante o treinamento dos Modelos de Entropia Máxima, é possível utilizar um dicionário de palavras raras para serem consideradas para a marcação sintática. Uma recomendação é o estudo mais aprofundado do funcionamento deste recurso para compreender as vantagens, desvantagens e suas outras prováveis utilidades. Durante a elaboração deste trabalho não foi possível encontrar esclarecimento suficiente na documentação do *SharpNLP* nem do *OpenNLP* sobre isto. Uma alternativa seria analisar o código fonte em C#, porém tal trabalho estaria além do escopo desta dissertação. Outra alternativa seria o aprofundamento teórico nos Modelos de Entropia Máxima buscando por maior compreensão sobre seu funcionamento e suas implementações em *software*.

7.4.12 CAPACIDADE DE RESPONDER PERGUNTAS

Durante os teste foi observado que o sistema apresenta uma reação significativa e útil, do ponto de vista de uma resposta, quando recebe instrução na forma de perguntas do tipo "Onde fica…" ou "Como chegar até …". Contudo, ele não foi dotado de recursos para efetivamente responder perguntas.

Recursos adicionais podem ser desenvolvidos com a possibilidade de reconhecer propriamente perguntas e formas de respondê-las. Uma primeira utilidade

para o usuário seria consultar as informações internas do sistema, com perguntas como "Quais os locais de referência que você conhece?" e "Quais os comandos que você pode executar?", entre outras.

7.4.13 NOVO CORPUS DE TREINAMENTO

Grande parte dos problemas que limitaram o reconhecimento dos diversos estilos possíveis de redação em língua natural é decorrente de insuficiência no *Corpus* de treinamento dos Modelos de Entropia Máxima, nos quais se baseia o *SharpNLP*. A elaboração de novo *Corpus*, ou a melhoria dos atualmente disponíveis, consistirá em apreciável contribuição, não só ao Sistema Robótico proposto, como principalmente aos futuros trabalhos e pesquisas que dependam do processamento estocástico da linguagem natural em Português.

Foram identificados os seguintes recursos do *Corpus* de treinamento em Português que podem ser melhorados no futuro:

- Marcação sintática de verbos conjugados no imperativo;
- Marcação sintática de verbos conjugados em diversificados tempos e pessoas (atualmente apresenta bastante terceira pessoa e os tempos passado e presente);
- Marcação sintática de verbos quando se usa sujeito oculto (dependendo do aprofundamento de estudos sobre a questão);
- Marcação sintática de números escritos na forma literal;
- Marcação sintática de abreviaturas; e
- Separação correta de sentenças quando da ocorrência de abreviaturas.

Tais melhorias de recursos tornarão o resultado obtido pelo processamento estatístico do Português mais consistente e próximo do limite teórico, tornando o desempenho da ferramenta *SharpNLP* tão útil como é para o Inglês.

7.4.14 FERRAMENTA PARA EDIÇÃO DO LÉXICO

Uma ferramenta que permita a edição do léxico poderá ser agregada como forma de facilitar a tarefa dos usuários e desenvolvedores que optem por utilizar o Sistema Robótico proposto para novos experimentos. A edição manual do léxico é trabalho enfadonho e propenso a mais erros quando executado manualmente.

Sendo recurso heurístico e parte essencial das funcionalidades de avaliação de contexto e extração de informações contidas nas instruções em linguagem natural,

seu conteúdo poderá ser mais convenientemente administrado e agilmente ajustado com recursos de uma ferramenta de edição planejada para tarefas como, por exemplo:

- Incluir e remover palavras;
- Incluir e remover contextos de interpretação (comandos do robô);
- Associar cada palavra aos diversos contextos de interpretação;
- Quantificar o grau de associação das palavras aos diversos contextos; e
- Parametrizar os valores computáveis de cada palavra sobre os contextos;

7.4.15 MÓDULO DE AJUDA AO USUÁRIO

Para auxiliar futuros usuários com a utilização dos recursos do Sistema Robótico e esclarecer suas funcionalidades e aplicações, será útil dispor de um módulo de ajuda. Ele poderá dispor da parte aplicável do conteúdo desta dissertação e outras informações pertinentes de acordo com a necessidade.

Este módulo de ajuda poderá também ser ampliado de acordo com as funcionalidades futuras que venham a ser desenvolvidas e agregadas ao sistema.

8. REFERÊNCIAS BIBLIOGRÁFICAS

ADALGEIRSSON, S. O. *Mebot, A Robotic Platform for Socially Embodied Telepresence*. Dissertação (Mestrado), MIT, 2009.

ALIBERT, K.; BRUEN, T. L. Telepresence: Harnessing the Human-Computer-Machine Interface. *Army Logistician, Professional Bulletin of United States Army Logistics*. Ft. Lee, EUA: Editor Army Logistician Almc, PB 700-06-06, v.3, n.6, 2006

ALLEN, J. *Natural Language Understanding*. Redwood City, CA: The Benjamin/Cummings Pub. Co., 1995.

ALLEN, S. Minotaur Pathfinder. 2008. Disponível em <http://archive.msdn.microsoft.com/pathfinding1>. Acessado em: 26 de agosto de 2012.

ALMEIDA, M. B.; BAX, M. P. Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. *Revista Ciência da Informação*. Brasília: s.n., set/dez de 2003.

ÁLVARES, A. J.; ROMARIZ, L. S. J. Jr. Desenvolvimento de um Manipulador com Dois Graus de Liberdade Controlado Remotamente Via Internet, *V Congresso de Engenharia Mecânica Norte e Nordeste*, Fortaleza, 27-30 de Outubro, pp. 529-536, 1998.

ÁLVARES, A. J.; ROMARIZ, L. S. J. Jr. Telerobotics: Methodology for the Development of a Through-the-internet Robotic Teleoperated System. *Revista Brasileira de Ciências Mecânicas*, v.24 pp.122–126, maio de 2002.

AMORETTI, M. S. M.; TAROUCO, L. M. R. Mapas Conceituais: modelagem colaborativa do conhecimento. Informática na Educação: Teoria & Prática, v.3, n.1, Porto Alegre: PGIE-UFRGS, 2000.

AMORETTI, M. S. M. Protótipos e estereótipos: aprendizagem de conceitos; Mapas Conceituais: experiência em Educação a Distância. *Informática na Educação: Teoria & Prática*, v.4, n.2, Porto Alegre: PGIE-UFRGS, 2001.

ARAÚJO, R. Computação Ubíqua: Princípios, Tecnologias e Desafios. *XXI Simpósio Brasileiro de Redes de Computadores*. Natal, UFRN e São Carlos, UFScar, 2003.

ARGILLANDER, J. *Maximum Entropy Modeling and Semantic Concept Detection. Department of Electrical and Communications Engineering, Helsinki University of Technology*, Dissertação de Mestrado, 2005.

BALDRIDGE, J.; MORTON, T.; BIERNER, G. OpenNLP. 2000. Disponível em <http://opennlp.apache.org>. Acessado em: 26 de agosto de 2012.

BARR, A.; FEIGENBAUM, E. A. *The Handbook of Artificial Intelligence*. v.I. Los Altos: Morgan Kaufmann, 1986. 423p.

BARROS, F. A.; ROBIN, J. Processamento de Linguagem Natural. *Jornada de Atualização em Informática, XVI Congresso da Sociedade Brasileira de Computação*. Recife: CIn UFPE, 1996.

BELLINGER, G.; CASTRO, D.; MILLS, A. *Data, Information, Knowledge, and Wisdom.* 2004. Disponível em <http://www.systems-thinking.org/dikw/dikw.htm>. Acesso em: 21 de abril de 2012.

BICK, E. *The Parsing System Palavras, Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework*. Tese (Doutorado) - Universidade de Århus, Dinamarca, 2000.

BLACK, P. E. *Finite State Machine - Dictionary of Algorithms and Data Structures (U.S. National Institute of Standards and Technology)*. 2008. Disponível em

<http://www.nist.gov/dads/HTML/finiteStateMachine.html> Acessado em: 15 de novembro de 2011.

BORST, P.; AKKERMAN, H.; TOP, J. Engineergin Ontologies. *International Journal of Human-Computer Studies*. [S.l.]: Academic Press, 1997. Vol. Volume 46.

BOUILLON, P. *Traitement Automatique des Langues Naturelles*. Bruxelas, Paris: Aupelf-Uref, Editions Duculot, 1998.

BURTON, R. R.; BROWN, J. S. *Semantic Grammar: A Technique for Constructing Natural Language Interfaces to Instructional Systems*. Bolt, Beranek & Newman, Inc., Cambridge, MA - Advanced Research Projects Agency (DOD), Washington, DC, EUA, Maio 1977.

CASARE, S. J. Uma ontologia funcional de reputação para agentes. Dissertação de Mestrado da Escola politécnica da USP. São Paulo: s.n., 2005.

CASTRO, S. *Ontologia*. Rio de Janeiro: Ed. Jorge Zahar, 2008.

CHEN, M. *Ontology and Semantic Web*. 2009. Disponível em <http://www.ecs.syr.edu/faculty/mccracken/nlpspring2009/Ontology.2009.pdf>. Acesso em: 27 de março de 2011.

CHOWDHURY, G. Natural Language Processing. *Annual Review of Information Science and Technology*, v.37, p.51-89, 2003.

CIBERNETICS. In: WIKIPÉDIA, a enciclopédia livre. Disponível em: <http://en.wikipedia.org/w/index.php?title=Cybernetics&oldid=419362687>. Acesso em: 15 de março de 2011.

CLARK, A.; FOX, C.; LAPPIN, S. et al. *The Handbook of Computations Linguistics and Natural Language Processing*. Oxford, UK: Wiley-Blackwell, 2010.

CLARK, P.; THOMPSON, J.; HOLMBACK, H.; DUNCAN, L. Exploiting a Thesaurus-Based Semantic Net for Knowledge-Based Search. *12th Conference on Innovative Applications of AI (AAAI / IAAI'2000)*, Austin, Texas, EUA, p.988-995, 2000.

DAHL, V.; FALL, A.; THOMAS, M. -C.; Driving robots through natural language. *IEEE International Conference on Intelligent Systems for the 21st Century*. v. 2, p. 1904-1908, Canadá, 1995.

ENGELS, S. *Maximum Entropy Markov Models*. Disponível em <http://www.cs.toronto.edu/~sengels/resources.html>, acessado em <24 de junho de 2011>.

FALBO, R. A.; RUY, F. B.; PEZZIN, J. ; MORO, R. D. Ontologias e Ambientes de Desenvolvimento de Software Semânticos. *Ibero-American Symposium on Software Engineering and Knowledge Engineering (JIISIC)*, 2004.

FELLBAUM, C. *WordNet: An Electronic Lexical Database*. Cambridge: MIT Press, 1998.

FERNANDES, H. N. *Português Descomplicado, Conforme o Novo Acordo Ortográfico*. Rio de Janeiro: Ed. Ferreira, 2010.

FILLMORE, C. The case for case. In Bach, E. and Harms, R.T. (orgs.), *Universals in linguistic theory*, pp. 1-88. Rinehard and Winston, New York, 1968a.

FILLMORE, C. Lexical Entries for Verbs. In *Foundations of Language*, pp. 373-393, 1968b.

FRANCONI, E. Description Logics for Natural Language Processing. In: Baader, F.; McGuinness, D. L.; Nardi, D.; Patel-Schneider, P. P. (editores) *Description Logics Handbook*. Cambridge: Cambridge University Press, 2001.

GANGEL, J. S. *A Motion Verb Interface to a Task Animation System*. Dissertação de Mestrado. School of Engineering and Applied Science, Universidade da Pensilvânia, Philadelphia, Pennsylvania, EUA, 1985.

GE, R.; MOONEY, R. Learning Semantic Parsers Using Statistical Syntactic Parsing Techniques. *Technical Report UT-AI-TR-06-327, Artificial Intelligence Lab.* Universidade do Texas em Austin, Fevereiro, 2006.

GELBUKH, A. Special issue: Natural Language Processing and its Applications. *Research in Computing Science*, v.46, s.n., México: Instituto Politécnico Nacional, Centro de Investigación en Computación, março, 2010.

GONZALES, M.; LIMA, V. L. S. Sintagma nominal em estrutura hierárquica temática na recuperação de informação. *XXI Congresso da Sociedade Brasileira de Computação – Encontro Nacional de Inteligência Artificial (ENIA)*, 2001.

GRANBERG, J.; GEORGSSON, F. *Learning Natural Language Interfaces over Expressive Meaning Representation Languages*. Dissertação (mestrado) - Universidade de Umeå, Suécia, 2010.

GRISHMAN, R. Computational linguistics: an introduction. *Studies in natural language processing*. Cambridge University Press, Cambridge, UK, 1986.

GUIZZARDI, G.; FALBO, R. A.; GUIZZARDI, R. S. S. A importância de Ontologias de Fundamentação para a Engenharia de Ontologias de Domínio, o caso do domínio de Processos de Software. *IEEE Latin America Transactions*, v. 6, n. 3, p. 244-251, 2008.

GUTHRIE, L.; PUSTEJOVKY, J.; WILKS, Y.; SLATOR, B. M. The Role of Lexicons in Natural Language Processing. *Communications of the ACM*, v.39, n.1, p.63-72, 1996.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics SSC, v. 4, n. 2*, p.100–107, 1968.

HEYLIGHEN, F.; JOSLYN, C.; MEYERS, R. A. (ed.). *Encyclopedia of Physical Science & Technology. 3ª ed, Academic Press*, New York, 2001.

HUNTER, A. *Semantic networks and WordNet in Knowledge Management*, 2000. Disponível em <http://www.cs.ucl.ac.uk/staff/a.hunter/tradepress/wordnet.html>. Acessado em 21 de junho de 2011.

HUSSE, C. Fast A-Star (2D) Implementation for C#. 2010. Disponível em <http://www.codeproject.com/Articles/118015/Fast-A-Star-2D-Implementation-for-C>. Acessado em: 26 de agosto de 2012.

JONG, H. K.; YONG, D. K.; KANG, H. L. The Third Generation of Robotics Ubiquitous Robot. *2nd International Conference on Autonomous Robots and Agents*, Palmerston North, New Zealand, 2004.

JURAFSKY, D.; MARTIN, J. *Speech and Language Processing – An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. New Jersey, USA: Prentice-Hall, 2000.

KASAMA, D. Y.; ZAVAGLIA, C.; ALMEIDA, G. M. B. Do termo à estruturação semântica: representação ontológica do domínio da Nanociência e Nanotecnologia utilizando a Estrutura Qualia. *Linguamática - Revista para o Processamento Automático das Línguas Ibéricas*, Portugal: Universidade do Minho, v.2, n.3, p.43–58, 2010.

KIM, J.; LEE, K.; KIM, Y. Ubiquitous Robot: A New Paradigm for Integrated Services. *2007 IEEE International Conference on Robotics and Automation*, Roma, Italy, p. 2853-2858, 2007.

KNIGHT, K.; RICH, E. *Inteligência Artificial*. 2ª ed. [S.l.]: McGraw-Hill, 1993. ISBN 155860221.

KOENIG, S.; LIKHACHEV, M. D* Lite, *Proceedings of the National Conference on Artificial Intelligence*, p. 476-483, 2002.

KONOPÍK, M. *Stochastic Semantic Analysis. Technical Report Nº DCSE/TR-2006-01, University of West Bohemia*, Pilsen, República Checa, 2006.

KROVETZ, R.; CROFT, W. B. Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems*, v. 10, n. 2, p. 115-141, 1992.

LAURIA, S.; BUGMANN, G.; KYRIACOU, T.; BOS, J.; KLEIN, E. Converting natural language route instructions into robot executable procedures. Proceedings of *11th IEEE International Workshop on Robot and Human Interactive Communication*, p. 223-228, UK, 2002.

LEVINE, R. I.; DRANG, D. G.; EDELSON, B. *Inteligência artificial e sistemas especialistas*. 1ª Ed. [S.l.: s.n.], 1988.

LEWIS, M. P. *Ethnologue: Languages of the World*, décima sexta edição. Dallas, Texas, EUA: SIL International (ed.), 2009.

LOPES, L. S.; TEIXEIRA, A. J. S.; RODRIGUES, M.; GOMES, D.; GIRÃO, J.; SÉNICA, N.; FERREIRA, L.; SOARES, P. A Robot with Natural Interaction Capabilities. Proceedings on *IEEE International Conference on Emerging Technologies and Factory Automation* – ETFA'2003, v. 1, p. 605-612, Portugal, 2003.

LOUKACHEVITCH, N. V.; SALLI, A. D.; DOBROV, B. V. Automatic Indexing Thesaurus Intended for Recognition of Lexical Cohesion in Texts. *NLDB'99 – 4th International Conference on Applications of Natural Language to Information Systems*. OCG Schriftenreihe Lecture Notes, v.129, p.203-208, Klagenfurt, Áustria, 1999.

MACKAY, D. *Path Planning with D*Lite Implementation and Adaptation of the D*Lite Algorithm, Technical Memorandum DRDC Suffield TM 2005-242*, Defence Research and Development Canada, DOI 10.1.1.67.6521, 2005.

MANNING, C.; KLEIN, D. 2003. *The Stanford Natural Language Processing Group: Stanford Classifier*. Disponível em < http://nlp.stanford.edu/software/classifier.shtml> Acessado em: 15 de novembro de 2011.

MCCALLUM, A.; FREITAG, D.; PEREIRA, F. Maximum Entropy Markov Models for Information Extraction and Segmentation. *Seventeenth International Conference on Machine Learning (ICML 2000)*, Stanford, CA, USA: Stanford University, ed. Pat Langley, Proceedings of the Seventeenth International Conference on Machine Learning, 2000.

MEL'CUK, I.; ZHOLKOVSKY, A. The Explanatory Combinatorial Dictionary. In: EVENS, Martha W. (Editora) *Relational Model of the Lexicon: Representing Knowledge in semantic networks*. New York: Cambridge University Press, 1992.

MICHAELIS: *dicionário prático da língua portuguesa*. São Paulo: Editora Melhoramentos, 2001.

MILLER, S., BOBROW, R., INGRIA, R., & SCHWARTZ, R. *Hidden understanding models of natural language. In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL-94)*, p. 25–32, 1994.

MORATZ, R.; TENBRINK, T. Natural Language Instructions for Joint Spatial Reference between Naive Users and a Mobile Robot. Proceedings on *11th IEEE International Workshop on Robot and Human Interactive Communication*, p. 229-234, Alemanha, 2002.

MORRIS, J. *Dijkstra's Algorithm - Data Structure and Algorithms*. 1998. Disponível em <http://www.cs.auckland.ac.nz/~jmor159/PLDS210/dijkstra.html>. Acessado em: 21 de abril de 2012.

NAIR, S. B.; PRASAD, P. B. A Real-Time Natural Language Command Interpreter for Robots. *IEEE International Conference on Computational Cybernetics and Simulation*, v. 4, P. 3972-3974, Índia, Outubro, 1997.

NETO, J. J. Tecnologia Adaptativa na Linguística Computacional. 2010. Disponível em <http://ccsl.ime.usp.br/files/Jo%C3%A3oJos%C3%A9Neto_TecnologiaAdaptativa.pdf>. Acessado em: 21 de abril de 2012.

NORTHEDGE, R. Statistical Parsing of English Sentences. 2006 Disponível em <http://www.codeproject.com/Articles/12109/Statistical-parsing-of-English-sentences> e <http://sharpnlp.codeplex.com>. Acessado em: 26 de agosto de 2012.

NOY, N. F.; MCGUINESS, D. L. Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 e Stanford Medical Informatics Technical Report SMI-2001-0880*. EUA: Stanford University, 2001.

NUNES, M. das G. V.; SILVA, B. C. D. da; RINO, L. H. M.; OLIVEIRA, O. N. de; MARTINS, R. T.; MONTILHA, G. *Introdução ao Processamento das Línguas Naturais*. Notas Didáticas do ICMC (Instituto de Ciências Matemáticas e de Computação), São Carlos, 1999.

OGATA, K. *Engenharia do Controle Moderno*. 4ª ed. São Paulo: Prentice Hall, 2003.

OTHERO, G. A. Linguística Computacional - princípios e aplicações. *I Workshop de Computação da Região Sul*. CD-ROM. Florianópolis: Unisul, 2004.

PEREIRA, S. do L. *Processamento da Linguagem Natural*. 2006. Disponível em: <http://www.ime.usp.br/~slago/IA-pln.pdf>. Acessado em: 26 de março de 2011.

PEREIRA, S. do L. *Processamento da Linguagem Natural*. 2011. Disponível em: <http://www.ime.usp.br/~slago/pl-12.pdf>. Acessado em: 26 de março de 2011.

PUSTEJOVSKY, J. *The Generative Lexicon*. Cambridge: The MIT Press, 1995.

SCHANK, R. *Conceptual Information Processing*. North-Holland Publishing Company, Holanda, 1975.

RATNAPARKHI, A. A Simple Introduction to Maximum Entropy Models for Natural Language Processing. Technical Report, EUA: Dept. of Computer and Information Science, University of Pennsylvania, 1997.

REZENDE, S. O. *Mineiração de Dados. XXV Congresso da Sociedade Brasileira de Computação*, São Lourenço, RS, p. 397-433, 2003.

RODRIGUES, L. F; SIMÕES, E. V; SALAZAR, S. R G. Ambiente de Programação de Robôs Móveis via Internet. *Anais do XXVI Congresso da SBC - EnRIIII*, p. 20 -29, 2006.

ROWLEY, J. *The wisdom hierarchy: representations of the DIKW hierarchy. Journal of Information Science*, n. 33 p. 163–180, UK, 2007.

RUSSEL, S. J.; RUSSEL, P. N. *Inteligência Artificial*. 2ª Ed. Rio de Janeiro: Elsevier, 2004.

SACCONI, L. A. Nossa Gramática: Teoria e Prática. São Paulo: Atual Editora, 1999.

SANTOS, A. dos; PELIZZONI, J. M. *Introdução ao Processamento da Linguagem Natural*. 2005. Disponível em <http://labic.icmc.usp.br/portugues/SIAE/default.htm>. Acesso em: 02 de setembro de 2005.

SCAPINI, I. K.; Relações entre Itens Lexicais. In: Poersch, J. M.; Wertheimer, A. M. C.; Ouro, M. E. P.; Ludwig, E. M.; Scapini, I. K.; Becker, B. F. Fundamentos de um Dicionário Remissivo. Florianópolis: 1o. Encontro do CELSUL, v.1, p.393-429, 1995.

SEARLE, J. R. What is a speech acts. *Oxford Readings in Philosophy*, p. 39-53. Oxford University Press, Londres, 1971.

SEDER, M.; PETROVI, I.; MAČEK, K. Motion Control of Mobile Robots in Indoor Dynamic Environment. *Scientific Literature Digital Library and Search Engine*, DOI: 10.1.1.133.9044, Croácia, Faculdade de Engenharia Elétrica e Computação, Universidade de Zagreb, 2008.

SELFRIDGE, M; VANNOY, W. A Natural Language Interface to a Robot Assembly System. *IEEE Journal of Robotics and Automation*, v. RA-2, n. 3, Setembro, 1986.

SONG, I.; GUEDEA, F.; KATRAY, F.; DAI, Y.; KHALIL, I. E. Natural Language Interface for Mobile Robot Navigation Control. Proceedings of the *2004 IEEE International Symposium on Intelligent Control*, p. 210-215, Taiwan, September, 2004.

SPARCK-JONES, K. *Synonymy and Semantic Classification*. Edinburgh: Edinburgh University Press, 1986.

SPECIA, L.; RINO, L. H. M. *Representação Semântica: Alguns Modelos Ilustrativos*. Relatório NILC-TR-02-12, Série de Relatórios do Núcleo Interinstitucional de Linguística Computacional. NILC - ICMC - USP, São Carlos, SP, Brasil, Julho, 2002.

SPILIOTOPOULOS, D.; ANDROUTSOPOULOS, I.; SPUROPOULOS, C. D. Human-Robot Interaction Based on Spoken Natural Language Dialogue. *European Workshop on Service and Humanoid Robots (Servicerob 2001)*, Grécia, 2001.

STENTZ, A. Optimal and Efficient Path Planning for Partially-Known Environments, *Proceedings of the International Conference on Robotics and Automation*, p. 3310–3317, 1994.

SWIGGER, K. M. *Symbolic Processing - Semantic Network*. 2010. Disponível em: <http://zeus.csci.unt.edu/swigger/csci3210/semantic.ppt> Acessado em: 19 de junho de 2011.

TRISTÃO, A. M. D.; FACHIN, G. R. B.; ALARCON, O. E. Sistema de classificação facetada e tesauros: instrumentos para organização do conhecimento. *Ciência da Informação* (revista), Brasília: Instituto Brasileiro de Informação em Ciência e Tecnologia - IBICT (ed.), v.33, n.2, p.161-171, maio/ago. 2004.

VACHASPATI, P.; WU, C. 2012. *Sentiment Classification using Machine Learning Techniques*. Massachussets Institute of Technology. Disponível em <http://pranjalv.com/sentiment/> Acessado em 25 de maio de 2012.

VIEIRA, R. Linguística computacional: fazendo uso do conhecimento da língua. *Entrelinhas*, ano 2, n. 4, São Leopoldo: UNISINOS, 2002.

VIEIRA, R.; LIMA, V. L. S. Linguística computacional: princípios e aplicações. *IX Escola de Informática da SBC-Sul*. Passo Fundo, Maringá, São José: Luciana Nedel (Ed.) SBC-Sul, 2001.

WALTER, S. M. *Natural Language Processing: A tutorial (revised)*. Relatório Interno RADC-TR-90-7, Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, NY, Janeiro 1990.

WERTHEIMER, A. M. C. O Dicionário Remissivo Comparado aos Outros Dicionários Existentes. In: Poersch, J. M.; Wertheimer, A. M. C.; Ouro, M. E. P.; Ludwig, E. M.; Scapini, I. K.; Becker, B. F. *Fundamentos de um Dicionário Remissivo*. Florianópolis: 1o. Encontro do CELSUL, v.1, p.393-429, 1995.

WIENER, N. *Cybernetics: Or Control and Communication in the Animal and the Machine*. Paris: Hermann & Cie e Cambridge, Massachusetts, EUA: MIT Press, 1948.

WILKS, Y. A.; SLATOR, B. M.; GUTHRIE, L. M. *Electric Words: Dictionaries, Computers and Meanings*. Cambridge: The MIT Press, 1996.

WINSTON, P. H. *Inteligência Artificial*. [s.l.]: Livros Técnicos e Científicos Editora Ltda., 1987.

WONG, Y. W.; MOONEY, R. J. Learning for Semantic Parsing Using Statistical Machine Translation Techniques. *Technical Report UT-AI-05-323, Artificial Intelligence Lab*. Universidade do Texas em Austin, Outubro, 2005.

ZADROZNY, B. *Processamento Estatístico de Linguagem Natural - Universidade Federal Fluminense*. 2008. Disponível em <http://www.ic.uff.br/~bianca/peln/> Acessado em: 15 de novembro de 2011.

ZHANG, L. *Maximum Entropy Modeling* - University of Edinburgh. 2007. Disponível em <http://homepages.inf.ed.ac.uk/lzhang10/maxent.html> Acessado em: 15 de novembro de 2011.

ZELEK, J. S. Human-Robot Interaction with a Minimal Spanning Natural Language Template for Autonomous & Tele-operated Control. Proceedings on the *1997 IEEE/RSJ International Conference on Intelligent Robots and Systems – IROS '97*, v. 1, p. 299-305, Setembro, 1997.

ZELENY, M. *Management support systems: towards integrated knowledge management, Human Systems Management* v. 7, p. 59–70, 1987.

A. APÊNDICE

A.1 O ROBÔ CURUMIM

O Curumim é um robô concebido e fabricado no país, destinado a fins educacionais e de pesquisa. Trata-se de um robô móvel inteligente, ou seja, entre outros recursos, ele dispõe de três eixos com rodas omnidirecionais acionados independentemente, cinco sensores de obstáculo que utilizam infravermelho, eletrônica embarcada e rádio de comunicação sem fio. É fabricado pela empresa de São Carlos – SP denominada *Extreme Robot,* ou *XBot*. Segundo afirma o fabricante, trata-se do primeiro robô móvel inteligente para as áreas de pesquisa e educação do Brasil.

Conforme consta da documentação do fabricante, é uma plataforma robótica e um ambiente para programação de robôs. Ao ser adquirido, é fornecido com o seguinte conjunto de componentes:

- Plataforma robótica: robô móvel;
- Rádio base com cabo USB;
- Duas baterias recarregáveis;
- Um carregador de baterias;
- Um transmissor embutido no robô e um receptor de vídeo externo; e
- *Software* para programação visual e convencional.

Como fornecida pelo fabricante, a plataforma robótica dispõe dos seguintes recursos integrados:

- Seis rodas omnidirecionais conectadas a três eixos independentes;
- Uma câmera de vídeo com transmissão sem fio;
- Cinco sensores de obstáculo que funcionam por infravermelho;
- Um acionamento de dispositivo;
- Um rádio para comunicação sem fio com computador;
- Uma antena flexível na parte superior; e
- A carcaça externa.

Uma visão geral do robô é mostrada na figura A.1.1.

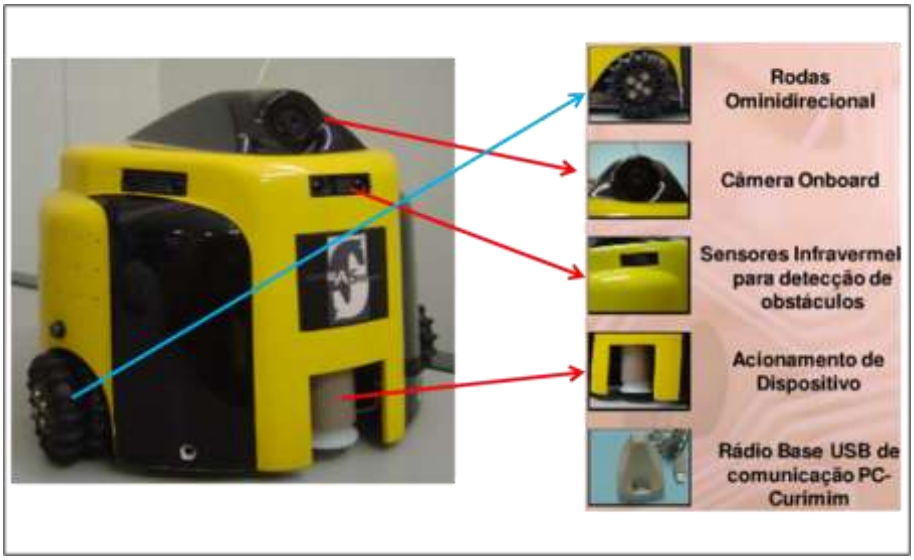


Figura A.1.1 – Visão geral do robô Curumim.

Fonte: *XBot – Extreme Robot* – São Carlos, SP

A.1.1 SISTEMA DE LOCOMOÇÃO

Seu sistema de locomoção consiste ao todo em três redutores e seis rodas omnidirecionais. Cada redutor é independente e possui duas rodas. Para assistir o momento, foram instalados *encoders* de 32 segmentos, que proporcionam resolução de 5mm. Esses *encoders* permitem que o sistema eletrônico embarcado possa medir a rotação do eixo das rodas.

As rodas omnidirecionais pesam cerca de 55g, segundo o fabricante. São montadas no robô duas a duas, acopladas uma à outra por uma estrutura rígida e própria denominada *flange*. O diâmetro total é de cerca de 50mm. Cada uma é composta por dezesseis pequenos anéis de borracha de 13mm de diâmetro cada. Essa construção permite que elas não ofereçam resistência ao momento no sentido do eixo. A figura A.1.2 ilustra a construção do par de rodas que é acoplado a cada eixo.

Os motorredutores são um conjunto que associa um motor elétrico acionado por corrente contínua, um *encoder* e uma caixa de redução. Sua principal função é reduzir a alta velocidade angular do motor para valores utilizáveis. Essa redução também permite um aumento no torque, ou seja, a força com que o conjunto pode promover o momento do robô.

A figura A.1.2 exibe também o aspecto externo do motorredutor.

Figura A.1.2 – Roda omnidirecional e motorredutor do Curumim.

Fonte: *XBot – Extreme Robot* – São Carlos, SP

Conforme diz o fabricante, o motor é acionado por 12Vcc, pode atingir a velocidade de 934 rpm e um torque de 0,4kgf.cm. A caixa de redução é de 1:13. O eixo para fixação das rodas possui 50mm de comprimento e 4mm de diâmetro. O peso total o conjunto é de cerca de 174g.

A figura A.1.3 mostra ilustrações do posicionamento de cada conjunto motorredutor-rodas em relação ao chassi da unidade móvel.

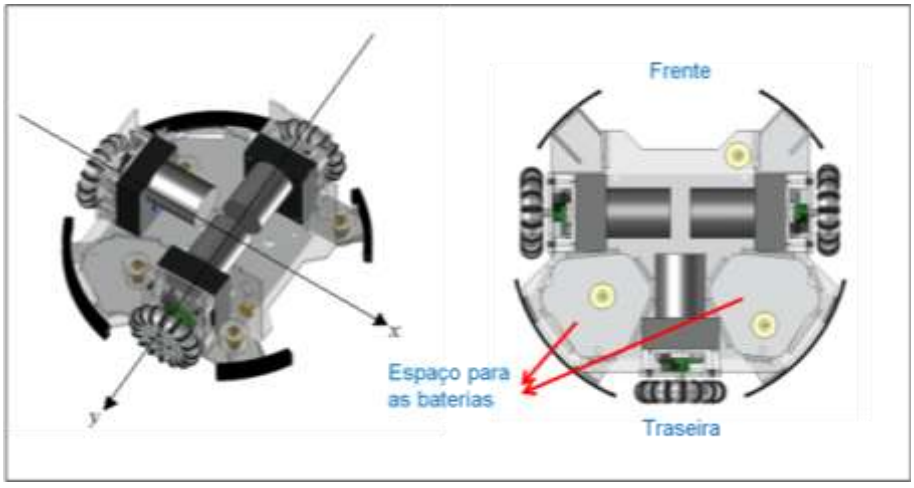


Figura A.1.3 – Posicionamento do conjunto motorredutor-roda no Curumim.

Fonte: *XBot – Extreme Robot* – São Carlos, SP

A.1.2 SENSORES INFRAVERMELHO

Para que a plataforma robótica seja capaz de detectar obstáculos, é dotada de cinco sensores que emitem sua própria radiação na faixa espectral do infravermelho. Funcionam pela detecção da reflexão ocasionada por obstáculos. Estão localizados ao redor da parte superior do robô. Conforme consta na documentação do fabricante, o alcance máximo é de aproximadamente 800m, com ângulo de ação de 5°. Segundo consta no manual do fabricante, o melhor resultado dos sensores se dá com obstáculos opacos.

Uma representação da posição relativa dos sensores de obstáculo e seus números de referência é mostrada na figura A.1.4.

Figura A.1.4 – Vista superior do Curumim e indicação dos sensores.

Fonte: *XBot – Extreme Robot* – São Carlos, SP

A figura A.1.5 mostra a vista frontal do Curumim, com destaque para a localização de três sensores, que são: 1- lateral frontal direita, 5 - frontal e 2 - lateral frontal esquerda.

A figura A.1.6 mostra a vista traseira do Curumim, com destaque para a localização de dois sensores, que são: 3 - lateral traseira esquerda e 4 - lateral traseira direita.

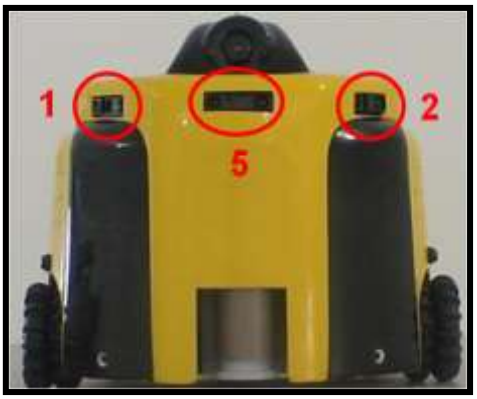


Figura A.1.5 – Vista frontal do Curumim e destaque para três sensores.
Fonte: *XBot – Extreme Robot* – São Carlos, SP

Figura A.1.6 – Vista traseira do Curumim e destaque para dois sensores.
Fonte: *XBot – Extreme Robot* – São Carlos, SP

A.1.3 CÂMERA COLORIDA

Para permitir uma realimentação visual ao usuário por meio de um receptor sem fio que pode ser conectado ao PC, monitor de vídeo ou televisor, uma câmera analógica colorida tipo *pinhole* (buraco de agulha) está instalada no topo da plataforma robótica. Ela opera de modo independente da eletrônica de controle da unidade móvel, sem que haja qualquer controle direto.

Por ser uma câmera tipo *pinhole* (buraco de agulha), possui foco e íris fixos proporcionados por um minúsculo buraco na sua parte frontal. Como informa o fabricante, é uma câmera de estado sólido, com sensor tipo CMOS, resolução de 380

linhas, transmissão via RF na faixa de frequência de 1,2GHz e permite um alcance de aproximadamente 200m.

A localização da câmera no robô e uma visão com mais detalhes são apresentados pela figura A.1.7.

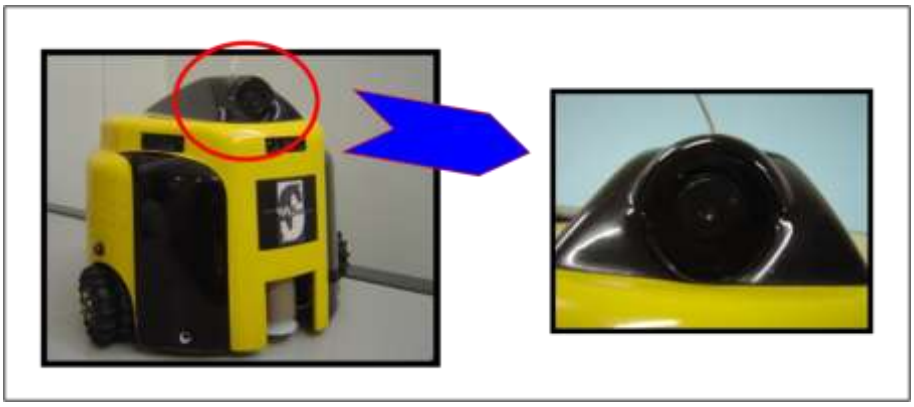


Figura A.1.7 – Posição da câmera no Curumim e a câmera em detalhe.
Fonte: *XBot – Extreme Robot* – São Carlos, SP

Seu receptor sem fio, também fornecido, permite imagens analógicas e possui saída de vídeo padrão RCA e áudio (não utilizado). O conector da antena é integrado ao chassi do receptor. Dispõe de um ajuste de sintonia final para a recepção. Para conexão com um PC, requer um dispositivo de captura ou digitalização de vídeo não fornecido. Como registra a documentação, o receptor é alimentado por ma tensão de 12V em corrente contínua, consome aproximadamente 250mA. É fornecido com um adaptador para a rede elétrica compatível com entrada de tensão de 85 a 265V em corrente alternada, tanto a 50 quanto a 60Hz, ou seja, é compatível com faixa de tensão ainda mais ampla que o bivolt comum (110/220V).

O aspecto geral do receptor de vídeo é apresentada pela figura A.1.8.

Figura A.1.8 – Receptor de vídeo para a câmera do Curumim.

Fonte: *XBot – Extreme Robot* – São Carlos, SP

A.1.4 PINÇA ACIONÁVEL

Um recurso de interação que foi adicionado à plataforma robótica é uma pinça acionável simples. Ela pode manipular objetos de pequeno porte, como uma caneta com a qual é possível criar desenhos à partir da trajetória do robô, por exemplo. Sensores de fim de curso permitem estabelecer a posição da pinça, que possui dois estados: abaixada e levantada.

Um detalhe da pinça é mostrado pela figura A.1.9.

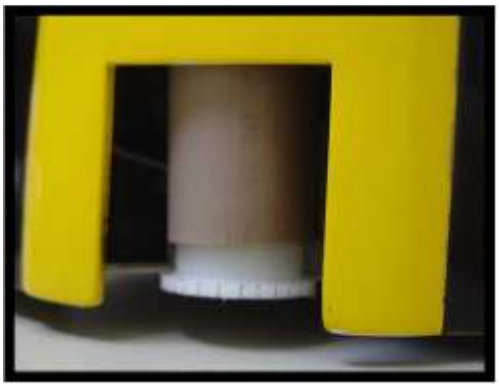


Figura A.1.9 – Detalhe da pinça incorporada no Curumim.

Fonte: *XBot – Extreme Robot* – São Carlos, SP

Ilustrações da utilização da pinça são apresentadas pela figura A.1.10.

Figura A.1.10 – Utilização da pinça.

Fonte: *XBot – Extreme Robot* – São Carlos, SP

A.1.5 ELETRÔNICA EMBARCADA

Para centralizar todo o controle da unidade móvel existe uma placa eletrônica embarcada. Segundo diz o fabricante, ela é baseada no microprocessador da *Texas Instruments* MSP430F149, é alimentada por uma tensão de 3,3V em corrente contínua e pesa cerca de 26g. A figura A.1.11 mostra seu aspecto geral.

Figura A.1.11 – Placa de controle, ou placa mãe, do Curumim.

Fonte: *XBot – Extreme Robot* – São Carlos, SP

A função da placa de controle, ou placa mãe, é reunir e organizar as informações provenientes dos sensores, tomar decisões baseadas nelas e permutar informações com o transceptor de rádio sem fio, com a placa de acionamento dos motores e ler os *encoders* das rodas. É apresentada na figura A.1.12, um diagrama de blocos da unidade móvel robótica.

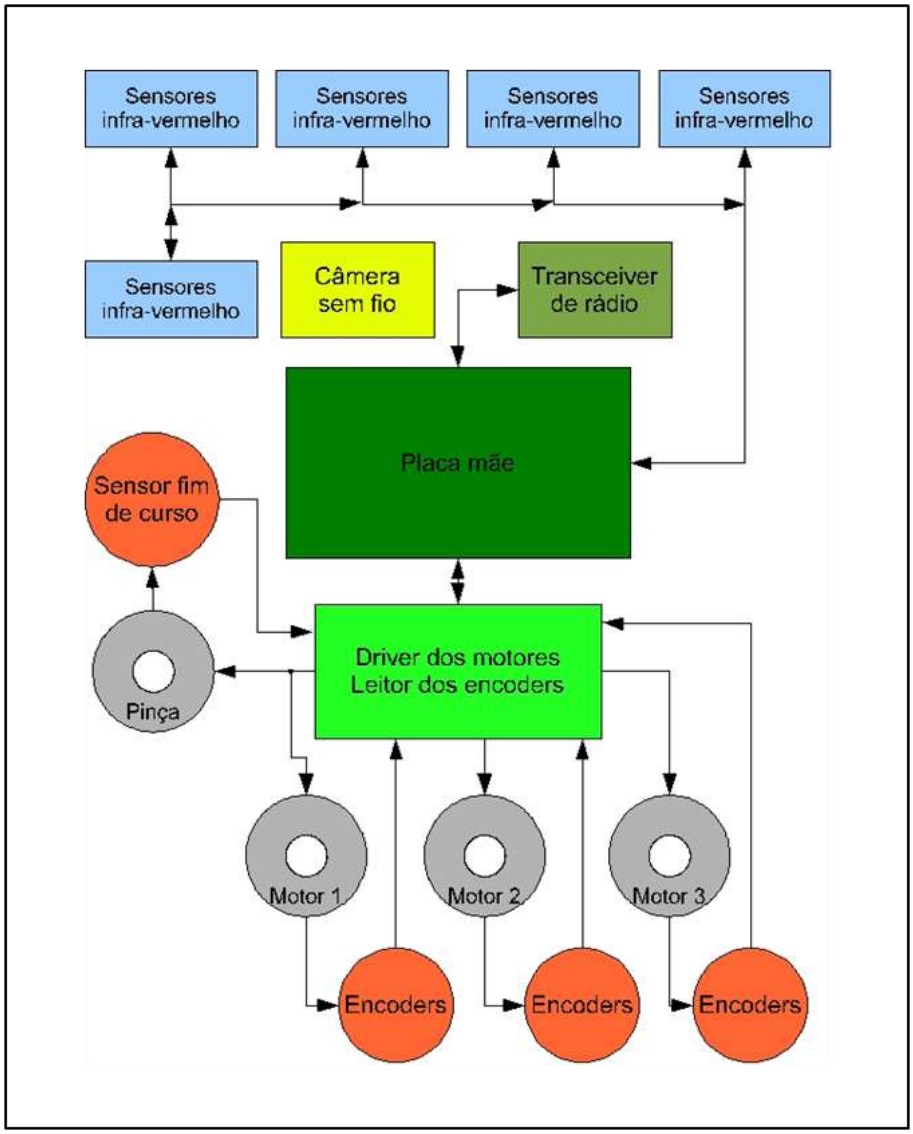


Figura A.1.12 – Diagrama de blocos da Unidade Móvel robótica.

Fonte: *XBot – Extreme Robot* – São Carlos, SP

A.1.6 RADIOBASE E COMUNICAÇÃO SEM FIO

A comunicação com o PC, e o comando do Curumim, é feita sem fio via rádio, através de uma unidade de Radiobase, que se conecta ao PC via interface USB. Esse transceptor digital, segundo a documentação do fabricante, opera na faixa dos 2,4GHz, recebe alimentação diretamente da interface USB, possui alcance de cerca de 75m em ambiente livre ou cerca de 20m em ambiente com obstáculo, possui velocidade de comunica bidirecional simétrica de 5kpbs, pesa cerca de 24g e possui dimensões máximas de 70 x 70 x 52 mm (A x C x L). Internamente, é controlada por um microcontrolador de fabricação da *Texas Instruments* MSP30, apresenta 125 canais distintos, é capaz de endereçar até 240 pontos, (por exemplo, robôs), incorpora uma antena e apresenta indicação de funcionamento por LED (diodo emissor de luz): um vermelho para dados recebidos e um verde para dados enviados. Visões dos aspectos externo e interno da radiobase são mostrados na figura A.1.13.

Figura A.1.13 – Radiobase ou transceptor de comunicação sem fio.

Fonte: *XBot – Extreme Robot* – São Carlos, SP

A.1.7 BATERIAS RECARREGÁVEIS E CARREGADOR

A energia da plataforma robótica é provida por um par de baterias recarregáveis, que podem ser removidas pela parte inferior da unidade móvel, por intermédio da remoção de uma tampa fixada por parafusos. Na documentação do Curumim, consta que a tensão da bateria em aberto (sem carga) é de cerca de 14,4V providos em corrente contínua máxima entre 2000mA e 2500mA, utiliza tecnologia de

NiMH (níquel-metal-hidreto) em composição de doze elementos e apresenta tempo estimado de utilização da unidade móvel em cerca de 30 minutos. O peso de cada bateria, conforme mostra a figura A.1.14, é de cerca de 400g e suas dimensões máximas são de 75 x 75 x 60 mm (A x C x L).

Figura A.1.14 – Baterias recarregáveis.
Fonte: *XBot – Extreme Robot* – São Carlos, SP

O carregador dessas baterias contém fonte de alimentação e conexão para recarga simultânea de duas baterias. Possui duas ventoinhas para refrigeração das baterias durante a carga, LED de sinalização de carregador ligado e um LED para cada bateria para indicação de carga em andamento ou carga concluída. Para uma noção do aspecto do carregador e sua fonte de alimentação, referir-se à figura A.1.15.

Figura A.1.15 – Carregador das baterias e sua fonte de alimentação.
Fonte: *XBot – Extreme Robot* – São Carlos, SP

A.1.8 POSSIBILIDADES DE CONTROLE DO CURUMIM

Não existem interfaces elétricas para acionamento manual. O *software* é o único intermediário possível entre o robô e o usuário. É por meio dele que o usuário poderá enviar os comandos para serem executados.

Com a plataforma robótica que constitui o Curumim, há três maneiras possíveis para utilização da interface de *software*:

1) Programação em blocos com o *Software* Curumim, desenvolvido pelo fabricante e fornecido como parte integrante do conjunto;
2) Programação visual com o MS *Robotics Studio*; e
3) Programação convencional utilizando bibliotecas de funções disponibilizadas pelo fabricante ou desenvolvidas especialmente.

O *Software* Curumim

O *Software* Curumim é uma interface de programação concebida e desenvolvida pelo próprio fabricante XBot. Permite programação simplificada para uso didático, com recurso de blocos funcionais e ajustes de parâmetros, de forma interativa. Essa interface também pode ser usada para iniciar programas em linguagem C ou C++ e que poderão ser desenvolvidos posteriormente conforme a necessidade do usuário.

Programação avançada

Um dos componentes de *software* fornecidos pela XBot para o Curumim é o conjunto de serviços do robô para o MS *Robotics Studio*. Com ele é possível utilizar os recursos de programação visual deste ambiente prático de desenvolvimento, inclusive a simulação virtual.

Igualmente componente de *software* são as bibliotecas e funções em linguagem de programação C e C++. Essas bibliotecas podem ser usadas, em conjunto com o ambiente de desenvolvimento integrado (IDE ou *Integraded Development Environment*) para criação de aplicações estruturadas de controle do robô.

Tais bibliotecas de funções permitem tanto o controle da comunicação através do transceptor ou radiobase, como também das funções da unidade móvel propriamente dita.

A linguagem de programação orientada para objetos C# também possui suporte pelo fabricante XBot. Entre os componentes fornecidos está um *wrapper* (recurso de interfaceamento específico de *software*) que permite a utilização da maioria

das funções disponíveis pela biblioteca destinada às linguagens C/C++ (execução não gerenciada de código de programa) na linguagem C# (execução gerenciada de código de programa).

Este último recurso tem maior foco ao longo desta dissertação, uma vez que a linguagem C# é utilizada para desenvolvimento do Sistema Robótico proposto.

Programação – funções disponíveis

A biblioteca de funções para utilização no desenvolvimento de aplicações disponibilizada pelo fabricante XBot dispõe de algumas funções que serão abordadas a seguir. Para organizar o entendimento, nesta dissertação, as funções foram agrupadas por semelhança.

As funções de movimentação são:

Andar Reto: provoca deslocamento no robô para frente ou para trás, a certa distância que deve ser informada e a certa velocidade que também deve ser informada. Apresenta-se como a função MOVE.

Rotacionar: provoca uma rotação do robô sobre seu próprio eixo em um sentido que deve ser informado se é para a esquerda ou para a direita, um ângulo em graus que deve igualmente ser informado e a certa velocidade que deve ser informada. Apresenta-se como a função ROTATE.

Fazer curva: provoca um deslocamento conjugado com rotação de forma que o robô executará uma curva mais ou menos aberta, para a direita ou para a esquerda e para frente ou para trás, conforme os parâmetros que forem informados. Nesta função devem ser informadas a distância que deve percorrer o motor da direita, a distância que deve percorrer o motor da esquerda, a velocidade que deve girar o motor da direita e a velocidade em que deve girar o motor da esquerda. Apresenta-se como a função TURN.

Ativar Motor: utilizada para acionar os motores da unidade móvel individualmente, um por vez ou em conjunto, para executar movimentos mais complexos. Nesta função devem ser informadas

a distância que deve percorrer o motor da direita, a distância que deve percorrer o motor da esquerda, a distância que deve percorrer o motor traseiro, a velocidade que deve girar o motor da direita, a velocidade em que deve girar o motor da esquerda e a velocidade em que deve girar o motor traseiro. Apresenta-se como a função TURN (a mesma anterior, porém com diferente número de parâmetros).

As funções relacionadas com a atuação e os sensores são:

Mover a pinça: provoca o deslocamento da pinça que deve ser informado e pode ser para cima ou para baixo. Apresenta-se na função TWIZZERS.

Status dos sensores: solicita uma comparação da distância lida pelo sensor com um valor informado. Nesta função devem ser informados o sensor desejado e a distância de referência. Apresenta-se na função SENSORS.

Outras funções diversas são:

Esperar: faz com que o robô permaneça parado por um tempo que deve ser informado. Apresenta-se na função SLEEP.

Aguardar conclusão: faz com que o *software* interno da unidade móvel aguarde a execução do último comando enviado e sinalize sua conclusão ao final. Apresenta-se na função WAITSTOPPED.

Programação – funções disponíveis pelo *wrapper* em C#

As funções disponíveis para utilização através da linguagem C# são:

- Andar Reto;
- Rotacionar;
- Ativar Motor;
- Mover a Pinça; e
- *Status* dos Sensores.

A.2 METODOLOGIA DE TESTE

A metodologia do experimento com usuários reais será descrita a seguir.

A.2.1 RECURSOS NECESSÁRIOS

- Um computador com o sistema robótico instalado; e
- O robô Curumim ligado e funcionando.

A.2.2 MEDIÇÕES E REGISTROS

Com vistas a avaliar o desempenho do Sistema Robótico proposto em interpretar as instruções em linguagem natural apresentadas pelos usuários que o experimentam, os seguintes dados são registrados automaticamente:

- Data e hora em que a instrução é submetida;
- A íntegra do texto da instrução;
- Cada oração da instrução conforme separada pelo processamento, já com as respectivas marcações sintáticas para as palavras;
- As probabilidades parciais e totais, para cada Esquema de Caso, resultante da avaliação de contexto de cada oração;
- Os comandos e mensagens de aviso e erro gerados com a interpretação das orações;
- O veredito do usuário, se há ou não erros, sobre a interpretação da instrução informada;
- A quantidade de erros, se houver;
- A quantidade total de comandos extraídos da instrução; e
- Registros sobre a execução dos comandos: tipo da execução (total, parcial ou passo a passo) e o número do passo da sequência de passos.

A.2.3 INSTRUÇÕES AO PESQUISADOR

O experimento é aplicado a um usuário por vez. Ela ou ele deve informar a instrução que deseja ver executada pelo robô em Português como normalmente faria a outro interlocutor.

Passo 01:

- Consultar se o usuário deseja participar da experiência;
- Caso afirmativo, acomodar o usuário em frete ao computador;
- Apresentar os recursos da Interface em Linguagem Natural necessários para a realização do experimento; e

- Narrar o texto padronizado “Descritivo da Experiência”.

Passo 02:

- Entregar-lhe o formulário contendo ambos: o texto padronizado “Descritivo da Experiência” e o “Termo de Esclarecimento e Livre Consentimento”;
- Solicitar que leia o formulário, preencha e assine o Termo;
- Entregar-lhe o “Questionário Demográfico” e solicitar que o leia e o preencha; e
- Não guardar os documentos na pasta ainda.

Passo 03:

- Consultar o usuário se está pronto para iniciar ou se necessita de algo;
- Caso afirmativo, colocar a Interface em Linguagem Natural no modo de teste e preencher o campo “Nome do Usuário” com as iniciais preenchidas no formulário “Questionário Demográfico” e o código “P1” justaposto às iniciais (em seguida);
- Informar o usuário que está livre para começar a Primeira Parte do experimento, ou seja, testar o sistema com instruções livres por dez (10) minutos;
- Guardar os formulários preenchidos na pasta;
- Depois de transcorrido este tempo, consultar o usuário se deseja testar livremente o sistema por mais um ciclo de dez minutos;
- Caso afirmativo, permitir que continue o teste livre por não mais de duas (2) vezes, totalizando trinta (30) minutos na Primeira Parte, e assim não se canse demais; e
- Encerrada a Primeira Parte da experiência, verificar se falta concluir o último teste (indicar se existem ou não erros de interpretação e clicar no botão de conclusão ou veredito).

Nota: O tempo total utilizado na Primeira Parte está registrado junto dos eventos da experiência.

Passo 04:

- Limpar o campo da instrução em Português e, no campo “Nome do Usuário”, substituir o código “P1” por “P2”;
- Consultar o usuário se está pronto para iniciar ou se necessita de algo;
- Caso afirmativo, poderá iniciar a Segunda Parte assim que ele desejar, que consiste em fazer o robô ir de um ponto inicial para um ponto final, ambos

previamente estabelecidos e indicados, com alguns obstáculos no caminho a serem contornados. Terá vinte e cinco (25) minutos para executar a tarefa;

- Transcorrido este tempo, consultar o usuário de necessita mais tempo para concluir a tarefa. Caso a tarefa seja concluída antes do tempo, submeter a frase-chave "Segunda parte concluída antes do tempo." a um ciclo de teste, para evidenciar o fato nos registros internos;
- Caso o usuário necessite mais tempo para concluir a tarefa, permitir que prossiga por mais quinze (15) minutos;
- Transcorrido este tempo, consultar o usuário de necessita mais tempo para concluir a tarefa;
- Caso o usuário necessite mais tempo para concluir a tarefa, permitir que prossiga por mais dez (10) minutos, totalizando um ciclo adicional de mais vinte e cinco (25) minutos. Informar o usuário que serão os últimos minutos restantes da experiência;
- Transcorrido este tempo, informar ao usuário sobre o final da experiência e solicitar que conclua o ciclo de teste, se for o caso; e
- A qualquer momento, assim que o usuário atingir o objetivo da tarefa, submeter a frase-chave "Segunda parte concluída." a um ciclo de teste, para evidenciar o fato nos registros internos.

Passo 05:

- Entregar o "Questionário sobre a experiência de uso" e solicitar que o preencha;
- Recolher o questionário preenchido e agradecer sua participação;
- Marcar no topo do questionário as iniciais do usuário conforme constante no campo "Nome do Usuário" da Interface em Linguagem Natural; e
- Guardar o questionário na pasta.

A.2.4 TAREFA DA SEGUNDA PARTE DA EXPERIÊNCIA

A segunda parte da experiência com os usuários estabelece uma tarefa específica que todos devem executar. Ela termina com o usuário atingindo seu objetivo ou com o término do tempo estabelecido. O registro automático dos eventos do sistema se encarregará dos tempos, que podem ser analisados após a parte prática.

O objetivo da tarefa específica desta segunda parte consiste em o usuário levar o robô primeiramente até um ponto "A" e depois para um ponto "B", a partir de um ponto inicial, todos previamente definidos. O espaço proposto para esta tarefa contém alguns obstáculos e alguns pontos de referência, também previamente estabelecidos.

Porém, os pontos A e B não são referências conhecidas do sistema, de forma que o usuário deve interagir com o robô, mediante o sistema, para conduzi-lo corretamente. A estratégia que utilizará é livre.

A figura A.2.1 ilustra a planta do espaço proposto para a tarefa específica, conforme vista na Interface em Linguagem Natural.

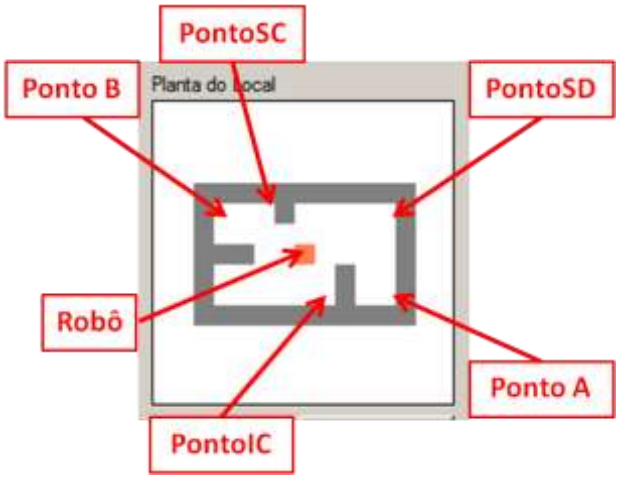


Figura A.2.1 – Planta do espaço proposto para teste com usuários.

O ponto inicial de localização do robô é no centro, uma das referências conhecidas do sistema. Os pontos de referência conhecidos pelo sistema e informados ao usuário são:

- Centro: local inicial do robô;
- PontoSD: (superior direito), localizado em um dos cantos do espaço;
- PontoIC: (inferior central), localizado próximo a um obstáculo; e
- PontoSC: (superior central), localizado próximo a outro obstáculo.

A montagem do espaço de experiência é mostrada em duas vistas distantas pelas figuras A.2.2 e A.2.3.

Figura A.2.2 – Montagem do espaço para experiência com usuários – Vista 1.

Figura A.2.3 – Montagem do espaço para experiência com usuários – Vista 2.

A.3 MODELOS DE DOCUMENTAÇÃO E QUESTIONÁRIO DE TESTE

A.3.1 DESCRITIVO DA EXPERIÊNCIA E TERMO DE ESCLARECIMENTO E LIVRE CONSENTIMENTO

Descritivo da Experiência

Obrigado por participar.

Durante este experimento será permitido a você controlar um robô através de um sistema baseado em software (programa de computador) através de instruções em Português do dia a dia, em linguagem natural, sem a necessidade de comandos técnicos específicos. Você deve usar o seu próprio jeito de dizer para controlar o robô, escrevendo essas instruções.

O que será avaliado neste experimento é a capacidade do sistema em interpretar e reconhecer as instruções para permitir que o robô execute o que foi solicitado. Com isto, será possível conhecer o que sistema já consegue executar e o que precisará ser ajustado para atender melhor os futuros usuários.

O sistema reconhecerá comandos de movimentação e rotação que o robô é capaz de executar, ou seja: deslocar para frente e para trás; girar para os lados e para trás; e ir até algum lugar conhecido ou pontos previamente nomeados.

Não há nenhum perigo ou risco de danos que possam ser causados a você ou a qualquer parte do sistema. O experimento é plenamente seguro. Algumas falhas aleatórias poderão ocorrer no programa ou no robô. Assim, tente prosseguir com o experimento ou, caso não seja possível prosseguir, peça ajuda ao pesquisador designado para conduzir os trabalhos.

Por fim, solicitamos sua autorização para usar os dados coletados sob a forma de questionário e registro automático de eventos, interno do software. Estes dados serão usados somente para elaboração de relatórios científicos. Sua privacidade é garantida e nenhum detalhe pessoal ou nome serão registrados, em momento algum.

A duração prevista é de cerca de vinte e cinco (25) minutos. Porém, você poderá se retirar a qualquer momento sem necessidade de se justificar, ou então prosseguir com os experimentos tanto tempo quanto julgar necessário. Ao final, pede-se que um questionário seja preenchido, para registrar suas observações sobre a experiência de uso do sistema e do robô.

Uma vez mais, somos gratos pela sua participação.

Termo de Esclarecimento e Livre Consentimento

Declaro que foram claramente esclarecidos para mim os procedimentos do experimento, seus objetivos, que minha identidade será respeitada e minha privacidade garantida. Compreendo que para a realização deste experimento serão feitos registros das minhas instruções e comandos, bem como será necessário tratamento posterior destes registros.

Declaro também que sei que poderei me retirar do experimento a qualquer momento.

Nome por extenso: _____.

São Paulo, ____/____/2012 Assinatura: _____.

A.3.2 QUESTIONÁRIO DEMOGRÁFICO

Questionário Demográfico							
Iniciais: _____ Gênero: [ ]M [ ]F Idade: _____ Grau de instrução: _____ Área de formação: _____							
Questões Por favor, quantifique seu grau de conhecimento, envolvimento ou uso em escala de 1 a 7.	**Baixo**	**Escala** Marque com "X" ou circule o número correspondente					**Alto**
Qual seu nível de conhecimento como usuário de programas de computador?	1	2	3	4	5	6	7
Qual seu nível de conhecimento como programador de computadores?	1	2	3	4	5	6	7
Qual seu nível de conhecimento sobre utilização de robôs	1	2	3	4	5	6	7
Qual seu nível de conhecimento sobre programação de robôs	1	2	3	4	5	6	7
Nível de conhecimento sobre a utilidade da robótica (para que serve, como é usada, quais as aplicações possíveis etc.)?	1	2	3	4	5	6	7
Qual seu nível de conhecimento sobre Inteligência Artificial ou Inteligência de Máquina?	1	2	3	4	5	6	7
Qual seu nível de conhecimento sobre sistemas capazes de compreender a Linguagem Natural ou o Português escrito?	1	2	3	4	5	6	7

A.3.3 QUESTIONÁRIO SOBRE A EXPERIÊNCIA DE USO

Questionário sobre a experiência de uso

Questões Por favor, quantifique o grau em escala de 1 a 7.	**Baixo**	**Escala** Marque com "X" ou circule o número correspondente					**Alto**
Como você avaliaria seu próprio desempenho durante o experimento?	1	2	3	4	5	6	7
Como você avaliaria o grau facilidade de uso do *software*?	1	2	3	4	5	6	7
Como você avaliaria o grau utilidade dos recursos do *software*?	1	2	3	4	5	6	7
Como você avaliaria o grau de resposta do robô aos comandos?	1	2	3	4	5	6	7
Qual o seu grau de facilidade para aprender a usar o sistema (*software* + robô)?	1	2	3	4	5	6	7
Qual o grau de facilidade para enviar instruções ao sistema (*software* + robô)?	1	2	3	4	5	6	7
Qual o seu grau de facilidade para executar os comandos enviados e reconhecidos pelo sistema (*software* + robô)?	1	2	3	4	5	6	7
Qual o grau de facilidade que a tarefa proposta apresentou?	1	2	3	4	5	6	7
Qual o seu nível geral de satisfação no uso do sistema?	1	2	3	4	5	6	7

Questões Por favor, marque uma das colunas: Sim ou Não. Use o espaço abaixo das perguntas para adicionar qualquer detalhar sua resposta.	**Sim**	**Não**
Você tentou utilizar formas diferentes de escrever a mesma instrução durante o experimento?		
Você tentou combinar vários comandos para compor instruções longas?		
Você conseguiu utilizar os três comandos básicos (movimentação, rotação e navegação) normalmente? Caso negativo, qual(is) teve mais dificuldade?		
Você acha que o sistema deixou de interpretar/reconhecer alguma instrução importante? Qual?		
Você acha que poderia refazer a tarefa proposta em menos tempo? Por quê? Como?		
Você gostaria de tentar novamente? Por quê?		
Você sentiu falta de alguma informação durante o experimento? Qual?		
Fez falta algum recurso ou capacidade do sistema durante o experimento? Qual(is)?		
Na sua opinião, qual foi a maior dificuldade para utilização do sistema (software e/ou robô)?		

A.4 TESTE DA COMUNICAÇÃO COM O ROBÔ

Observou-se que a comunicação sem fio com o robô apresentava algumas falhas ocasionais, de forma que o robô não executava os comandos enviados pelo sistema, ou mesmo parava de responder aos comandos enviados. Neste último caso, a solução consistia em desligar e religar o robô.

A seguir será relatado um experimento utilizado para avaliar a comunicação sem fio entre o sistema e o robô.

A.4.1 OBJETIVO DO EXPERIMENTO

O objetivo é identificar a frequência de falhas na comunicação em função da distância entre o robô e a radiobase. Buscando avaliar possíveis interferências de rádio, o experimento foi executado duas vezes, em horários diferentes.

A.4.2 DESCRIÇÃO DO EXPERIMENTO

Para as distâncias de um a cinco metros a partir da radiobase, em pontos distantes um metro cada, o robô será posicionado e um dentre um conjunto escolhido de comandos será executado trinta vezes para observar se o robô recebe e executa esse comando.

Cada falha de comunicação será adicionada ao total até o final das trinta vezes que cada comando é enviado ao robô. Ou seja, ao final, será possível calcular uma porcentagem de falhas observadas em relação às trinta vezes que o comando é enviado.

Os que foram escolhidos para o conjunto de comandos são:

- Mover(100,100): Mover 10 cm na velocidade 100;
- Rotacionar (90,100): Girar 90 graus a esquerda na velocidade 100; e
- Ler Sensores: sequencia completa de leitura dos sensores, conforme método implementado no Sistema Robótico.

As figuras A.4.1 e A.4.2 exibem duas vistas distintas do local da montagem.

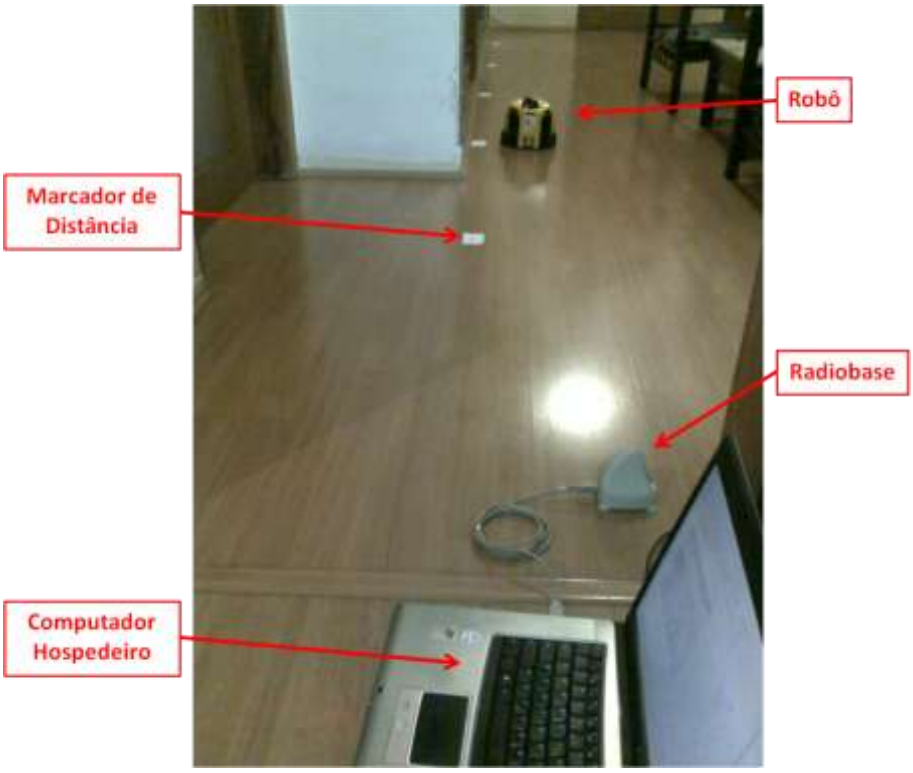


Figura A.4.1 – Montagem do experimento para avaliar a comunicação sem fio.

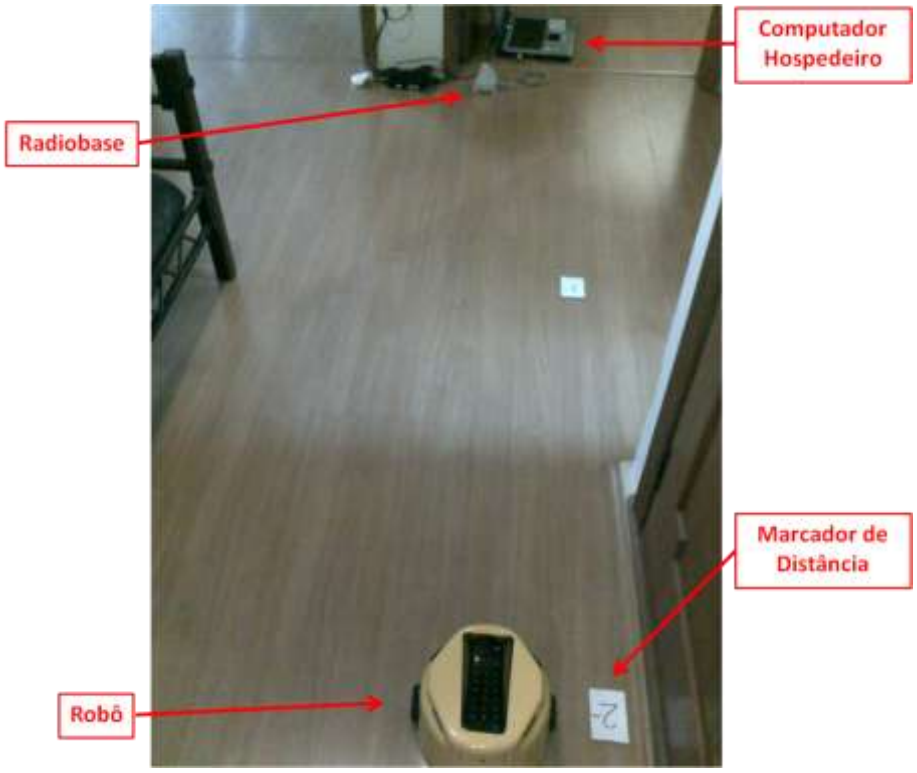


Figura A.4.2 – Outra vista do experimento para avaliar a comunicação sem fio.

A.4.2 RESULTADO DO EXPERIMENTO

Primeira sessão de testes:

A tabela A.4.1 exibe os resultados obtidos na primeira vez que o experimento foi executado, num feriado, sexta-feira, às 9h, portanto de manhã. Os números indicam a quantidade de erros observados para cada ciclo de trinta vezes que cada comando foi enviado ao robô.

Tabela A.4.1 – Resultado da primeira avaliação da comunicação sem fio.

Quant. de Erros	Distância (m)				
Comando	1	2	3	4	5
Mover (100,100)	0	0	4	0	2
Rotacionar (90,100)	0	0	0	0	4
Ler sensores	0	0	1	0	2

Os valores da tabela A.4.1 podem ser exibidos na forma de porcentagem, conforme mostra a tabela A.4.2.

Tabela A.4.2 – Valores da tabela A.4.1 recalculados em porcentagem.

% de Erros	Distância (m)				
Comando	1	2	3	4	5
Mover (100,100)	0,00%	0,00%	13,33%	0,00%	6,67%
Rotacionar (90,100)	0,00%	0,00%	0,00%	0,00%	13,33%
Ler sensores	0,00%	0,00%	3,33%	0,00%	6,67%

Análise dos resultados da primeira sessão:

É possível observar que o comando Mover foi o primeiro a ser afetado por problemas com a comunicação sem fio a uma distância de três metros. Nessa mesma distância, foi observada também interferência na leitura dos sensores.

Porém, não se observou nenhuma dessas duas falhas na distância de quatro metros. E elas voltaram a aparecer na distância de cinco metros.

Segunda sessão de testes:

A segunda vez que o experimento foi conduzido, foi no sábado seguinte, as 22h, portanto de noite. Os números da tabela A.4.3 exibem a quantidade de falhas para cada ciclo de trinta vezes que cada comando foi enviado.

Tabela A.4.3 – Resultado da segunda avaliação da comunicação sem fio.

Quant. de Erros	Distância (m)				
Comando	1	2	3	4	5
Mover (100,100)	0	0	0	1	1
Rotacionar (90,100)	0	0	0	1	2
Ler sensores	0	0	0	1	1

Recalculando os valores da tabela A.4.3 em forma de porcentagem, o resultado é mostrado na tabela A.4.4.

Tabela A.4.4 – Valores da tabela A.4.3 recalculados em porcentagem.

% de Erros	Distância (m)				
Comando	1	2	3	4	5
Mover (100,100)	0,00%	0,00%	0,00%	3,33%	3,33%
Rotacionar (90,100)	0,00%	0,00%	0,00%	3,33%	6,67%
Ler sensores	0,00%	0,00%	0,00%	3,33%	3,33%

Análise dos resultados da segunda sessão:

De maneira interessante, as falhas observadas na distância de três metros de distância não mais surgiram. Contudo, foram observadas falhas na distância de quatro metros.

As falhas observadas na primeira vez para cinco metros foram observadas novamente, porém e menor frequência.

Considerações finais sobre o experimento:

Saber exatamente a causa das falhas e seu comportamento variado em função do horário está além do escopo deste trabalho. Por razões práticas, considerou-se que uma distância segura para realização dos experimentos com o robô seria três (3) metros.

Porém, deve-se observar que não podem ser descartadas outras fontes de falha de comunicação, como as oriundas do funcionamento interno do robô, da radiobase e da comunicação via USB entre a radiobase e o computador hospedeiro.

A.5 TESTES E RESULTADOS DURANTE O DESENVOLVIMENTO

A.5.1 TESTE 1 – CONHECENDO AS FUNÇÕES DO SHARPNLP

O objetivo é identificar as funções disponíveis na biblioteca do *SharpNLP*, compreender suas propriedades e relacionar aquelas que podem ser úteis ao Sistema Robótico.

Em seguida, testar essas funções com texto em linguagem natural em inglês, vinculando-se a comandos do robô Curumim.

Funções disponíveis, suas propriedades e suas utilidades:

Segundo a documentação do *SharpNLP*, as funções disponíveis para processamento da linguagem natural em língua inglesa são:

- *Sentences Detect* (Detecta Sentenças) – divide o texto em sentenças ou frases nos pontos mais prováveis;
- *Tokenize* ("Tokeniza" ou divide em *tokens*) – separa uma sentença ou frase, previamente separada pela função anterior, em palavras ou *tokens,* da maneira mais provável possível;
- *POS Tagger* (Etiquetador de Classes Gramaticais) – marca cada palavra ou *token,* previamente separado pela função anterior, com sua mais provável classe gramatical conforme utilização na sentença ou frase em questão;
- *Chunker* (Agrupador de Sentenças) – utilizado para agrupar conjuntos de palavras ou *token* (trechos de sentença) por função sintática;
- *Parser* (Analisador de Sentenças) – analisa sintaticamente a sentença ou frase, indicando por complete os agrupamentos e classes gramaticais de cada estrutura que a compõe; e
- *NameFind* (Encontra Nomes) – marca a identificação de nomes mais prováveis presentes na sentença ou frase, que podem ser: datas, locais, moedas, organizações, porcentagem, pessoas e tempo.

Utilizando uma miniaplicação veiculada junto do *SharpNLP*, é possível experimentar a funcionalidade dessas funções, sendo que é convenientemente sugerido um texto em inglês ao ser iniciada (que pode ser substituído por outro do usuário a qualquer momento). A figura A.5.1 mostra a tela inicial.

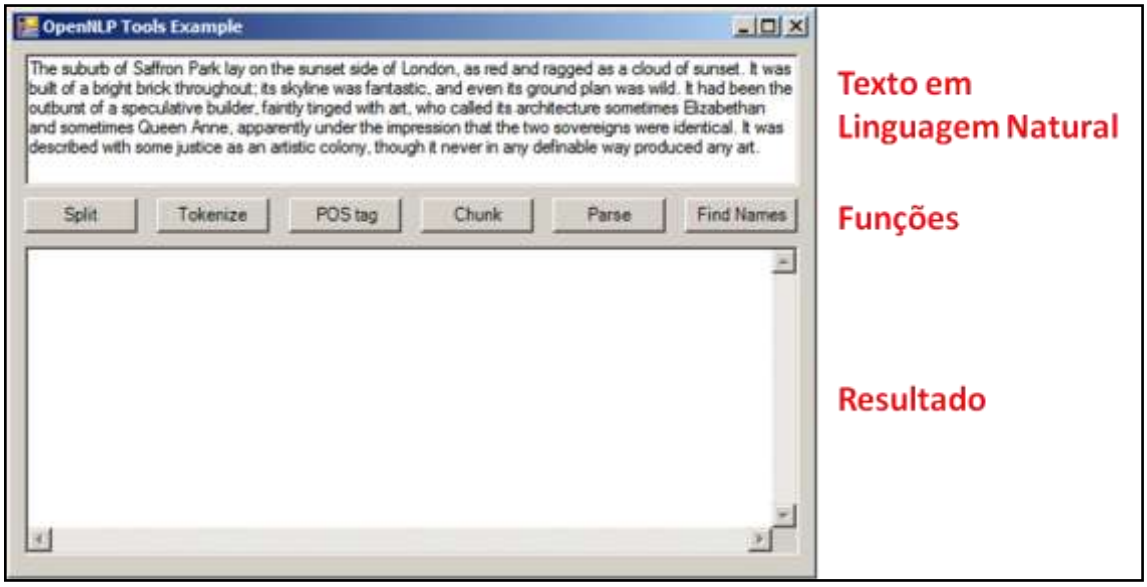

OpenNLP Tools Example
The suburb of Saffron Park lay on the sunset side of London, as red and ragged as a cloud of sunset. It was built of a bright brick throughout; its skyline was fantastic, and even its ground plan was wild. It had been the outburst of a speculative builder, faintly tinged with art, who called its architecture sometimes Elizabethan and sometimes Queen Anne, apparently under the impression that the two sovereigns were identical. It was described with some justice as an artistic colony, though it never in any definable way produced any art.
Split
Tokenize
POS tag
Chunk
Parse
Find Names
Texto em Linguagem Natural
Funções
Resultado

Figura A.5.1 – Tela inicial da miniaplicação de teste do *SharpNLP*.

Ao clicar-se no botão "*Split*", a função *SentenceDetect* é executada e o resultado obtido é a separação das sentenças do texto inicial. O aspecto é mostrado na figura A.5.2.

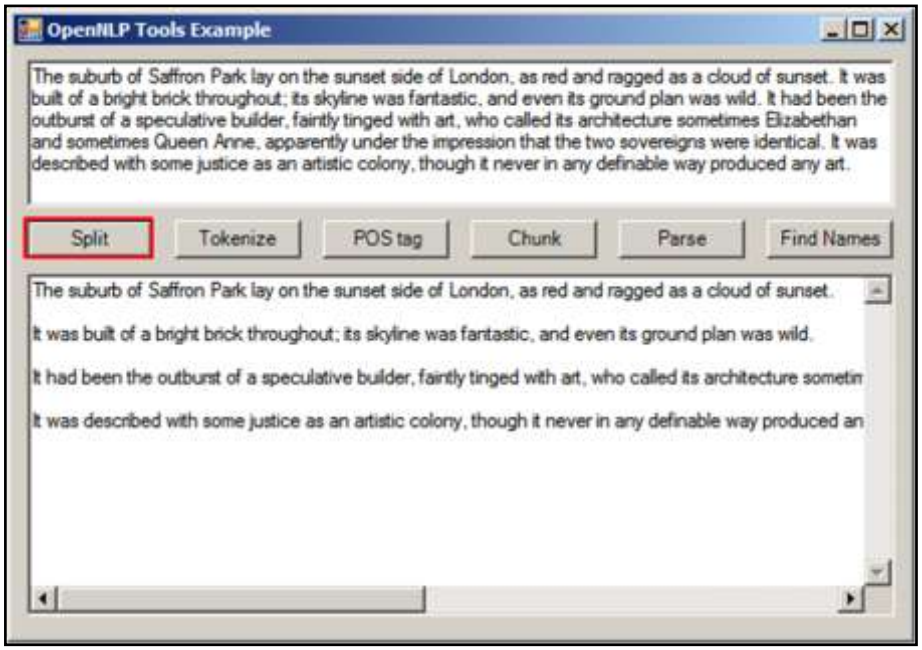

OpenNLP Tools Example
The suburb of Saffron Park lay on the sunset side of London, as red and ragged as a cloud of sunset. It was built of a bright brick throughout; its skyline was fantastic, and even its ground plan was wild. It had been the outburst of a speculative builder, faintly tinged with art, who called its architecture sometimes Elizabethan and sometimes Queen Anne, apparently under the impression that the two sovereigns were identical. It was described with some justice as an artistic colony, though it never in any definable way produced any art.
Split
Tokenize
POS tag
Chunk
Parse
Find Names
The suburb of Saffron Park lay on the sunset side of London, as red and ragged as a cloud of sunset.
It was built of a bright brick throughout; its skyline was fantastic, and even its ground plan was wild.
It had been the outburst of a speculative builder, faintly tinged with art, who called its architecture sometin
It was described with some justice as an artistic colony, though it never in any definable way produced an

Figura A.5.2 – Resultado da função *SentenceDetect* pela miniaplicação do *SharpNLP*.

O botão "*Tokenize*" executa a função homônima e o resultado é a separação das palavras e *tokens* em cada sentença do texto inicial, conforme mostra a figura A.5.3.

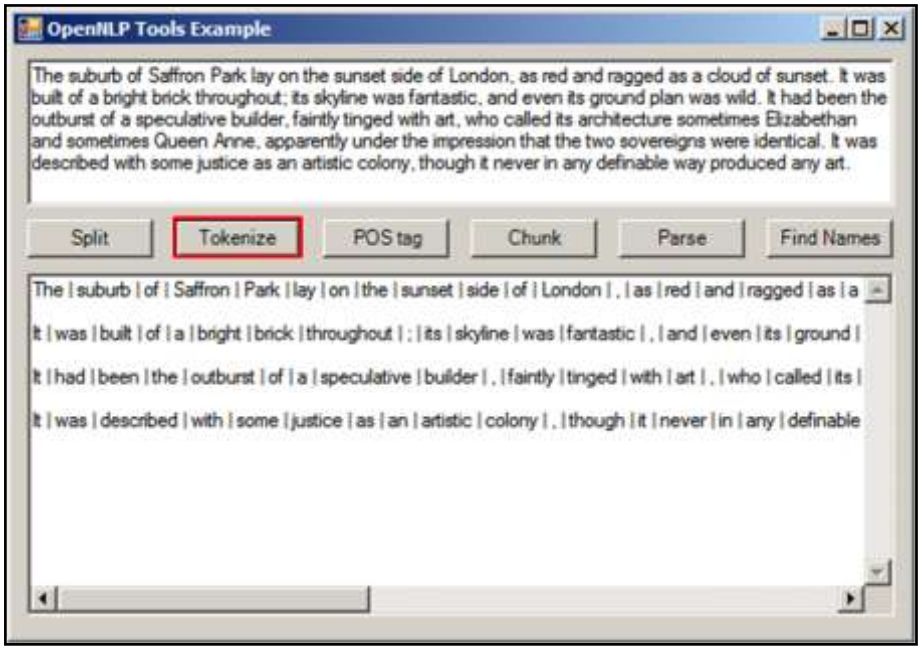

OpenNLP Tools Example
The suburb of Saffron Park lay on the sunset side of London, as red and ragged as a cloud of sunset. It was built of a bright brick throughout; its skyline was fantastic, and even its ground plan was wild. It had been the outburst of a speculative builder, faintly tinged with art, who called its architecture sometimes Elizabethan and sometimes Queen Anne, apparently under the impression that the two sovereigns were identical. It was described with some justice as an artistic colony, though it never in any definable way produced any art.
Split
Tokenize
POS tag
Chunk
Parse
Find Names
The | suburb | of | Saffron | Park | lay | on | the | sunset | side | of | London | , | as | red | and | ragged | as | a
It | was | built | of | a | bright | brick | throughout | ; | its | skyline | was | fantastic | , | and | even | its | ground |
It | had | been | the | outburst | of | a | speculative | builder | , | faintly | tinged | with | art | , | who | called | its |
It | was | described | with | some | justice | as | an | artistic | colony | , | though | it | never | in | any | definable

Figura A.5.3 – Resultado da função *Tokenize* pela miniaplicação do *SharpNLP*.

O botão "*POS Tag*" dispara a função *PosTagger* e cada palavra ou *token* recebe marcação da sua classe gramatical mais provável, como mostrado na figura A.5.4.

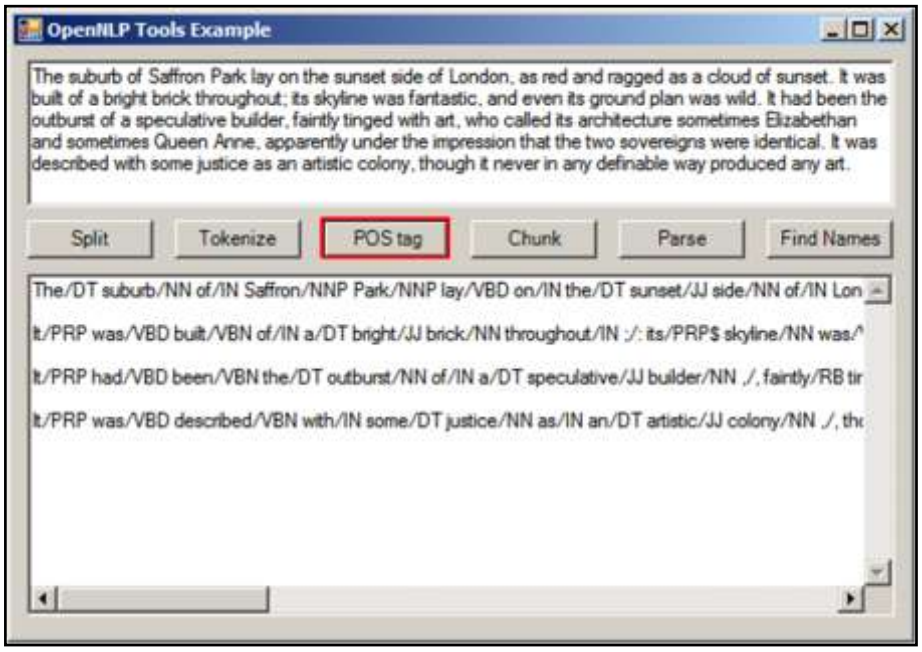

OpenNLP Tools Example
The suburb of Saffron Park lay on the sunset side of London, as red and ragged as a cloud of sunset. It was built of a bright brick throughout; its skyline was fantastic, and even its ground plan was wild. It had been the outburst of a speculative builder, faintly tinged with art, who called its architecture sometimes Elizabethan and sometimes Queen Anne, apparently under the impression that the two sovereigns were identical. It was described with some justice as an artistic colony, though it never in any definable way produced any art.
Split
Tokenize
POS tag
Chunk
Parse
Find Names
The/DT suburb/NN of/IN Saffron/NNP Park/NNP lay/VBD on/IN the/DT sunset/JJ side/NN of/IN Lon
It/PRP was/VBD built/VBN of/IN a/DT bright/JJ brick/NN throughout/IN ;/: its/PRP$ skyline/NN was/^
It/PRP had/VBD been/VBN the/DT outburst/NN of/IN a/DT speculative/JJ builder/NN ,/, faintly/RB tir
It/PRP was/VBD described/VBN with/IN some/DT justice/NN as/IN an/DT artistic/JJ colony/NN ,/, th

Figura A.5.4 – Resultado da função *PosTagger* pela miniaplicação do *SharpNLP*.

O botão "*Chunk*" executa a função *Chunker*, agrupando as orações de cada sentença do texto inicial pela sua função sintática, conforme exibe a figura A.5.5.

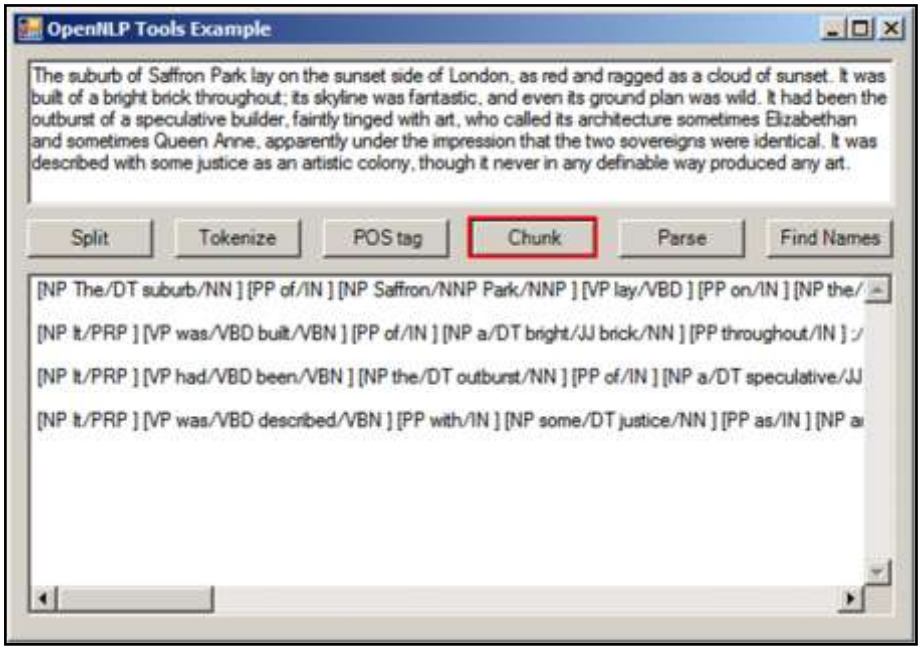

OpenNLP Tools Example
The suburb of Saffron Park lay on the sunset side of London, as red and ragged as a cloud of sunset. It was built of a bright brick throughout; its skyline was fantastic, and even its ground plan was wild. It had been the outburst of a speculative builder, faintly tinged with art, who called its architecture sometimes Elizabethan and sometimes Queen Anne, apparently under the impression that the two sovereigns were identical. It was described with some justice as an artistic colony, though it never in any definable way produced any art.
Split
Tokenize
POS tag
Chunk
Parse
Find Names
[NP The/DT suburb/NN] [PP of/IN] [NP Saffron/NNP Park/NNP] [VP lay/VBD] [PP on/IN] [NP the/
[NP It/PRP] [VP was/VBD built/VBN] [PP of/IN] [NP a/DT bright/JJ brick/NN] [PP throughout/IN]
[NP It/PRP] [VP had/VBD been/VBN] [NP the/DT outburst/NN] [PP of/IN] [NP a/DT speculative/JJ
[NP It/PRP] [VP was/VBD described/VBN] [PP with/IN] [NP some/DT justice/NN] [PP as/IN] [NP a

Figura A.5.5 – Resultado da função *Chunker* pela miniaplicação do *SharpNLP*.

Ao clicar-se no botão "*Parse*", a função *Parser* é executada e o resultado obtido é a análise sintática completa em inglês das sentenças, conforme a figura A.5.6.

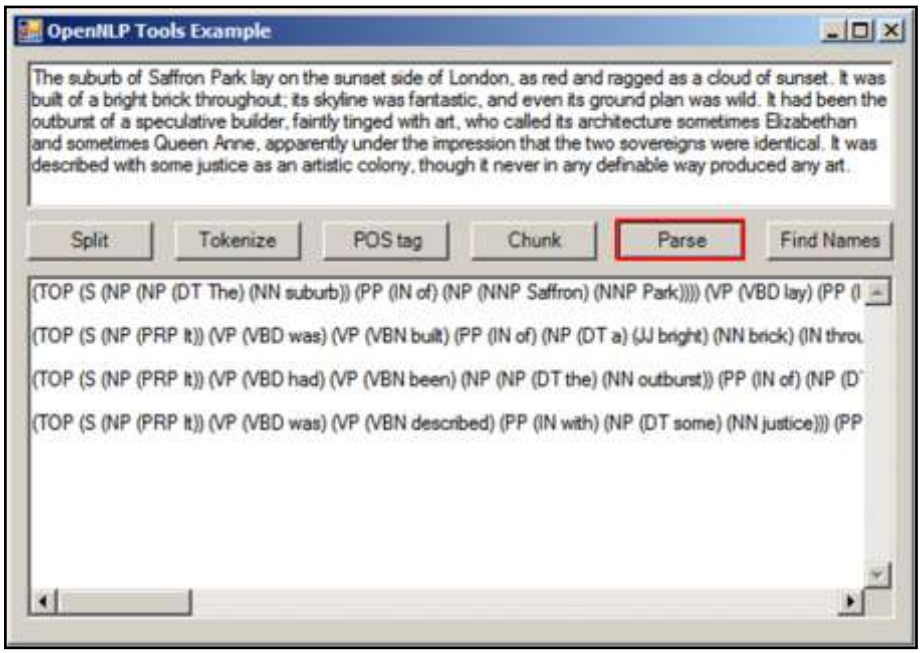

OpenNLP Tools Example
The suburb of Saffron Park lay on the sunset side of London, as red and ragged as a cloud of sunset. It was built of a bright brick throughout; its skyline was fantastic, and even its ground plan was wild. It had been the outburst of a speculative builder, faintly tinged with art, who called its architecture sometimes Elizabethan and sometimes Queen Anne, apparently under the impression that the two sovereigns were identical. It was described with some justice as an artistic colony, though it never in any definable way produced any art.
Split
Tokenize
POS tag
Chunk
Parse
Find Names
(TOP (S (NP (NP (DT The) (NN suburb)) (PP (IN of) (NP (NNP Saffron) (NNP Park)))) (VP (VBD lay) (PP (I
(TOP (S (NP (PRP It)) (VP (VBD was) (VP (VBN built) (PP (IN of) (NP (DT a) (JJ bright) (NN brick) (IN throu
(TOP (S (NP (PRP It)) (VP (VBD had) (VP (VBN been) (NP (NP (DT the) (NN outburst)) (PP (IN of) (NP (D
(TOP (S (NP (PRP It)) (VP (VBD was) (VP (VBN described) (PP (IN with) (NP (DT some) (NN justice))) (PP

Figura A.5.6 – Resultado da função *Parser* pela miniaplicação do *SharpNLP*.

Clicando-se no botão "*Find Names*", a função *NameFind* é executada, resultando na indicação dos nomes reconhecidos, conforme mostrado na figura A.5.6.

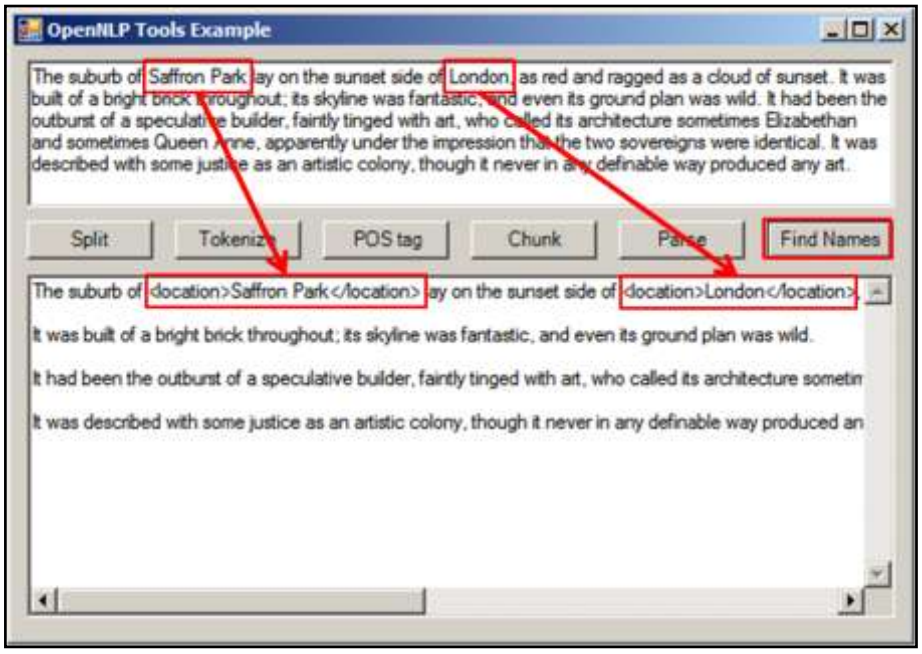

OpenNLP Tools Example
Split
POS tag
Chunk
Find Names
It was built of a bright brick throughout; its skyline was fantastic, and even its ground plan was wild.
It had been the outburst of a speculative builder, faintly tinged with art, who called its architecture sometim
It was described with some justice as an artistic colony, though it never in any definable way produced an

Figura A.5.7 – Resultado da função *NameFind* pela miniaplicação do *SharpNLP*.

Análise e Conclusão Parciais:

Tendo em vista as etapas necessárias para o processamento da linguagem natural conforme esclarecem Barros e Robin (1996) e as funções disponíveis no *SharpNLP*, ou seja, não dispõe daquelas capazes de processamento semântico, pragmático ou de discurso. Assim, conforme os testes realizados as três funções que podem ser utilizadas para a primeira parte do processamento do texto para utilização no Sistema Robótico são:

- *SentencesDetect* (Separa Sentenças);
- *Tokenize* ("Tokeniza" Sentença – separa palavras); e
- *PosTagger* (Etiqueta Classe Gramatical).

Buscando saber se as demais funções seriam úteis para o Sistema Robótico, concluiu-se que não será preciso conhecer a estrutura sintática do texto em sua totalidade. Tendo-se em vista que o objetivo é a interpretação do texto para extração de

comandos e parâmetros executáveis pelo robô, o nível de profundidade do processamento sintático é suficiente conhecer as marcações das classes sintáticas. Estas marcações indicam a função das palavras na estrutura das frases, sendo possível extrair o sentido de grande parte das instruções possível em linguagem natural, uma vez conhecendo o significado das palavras, à exceção das formas e estilos mais complexos tais como: como orações coordenadas, uso de correferências (frases escritas com palavras como “este”, “aquele”, “ele” etc.) e menções que dependem de memória de contexto.

Também, é necessário considerar que os modelos utilizados até aqui funcionam somente na língua inglesa. Foi então pesquisada a existência de modelos apropriados em Português, para que essas funções realizem o processamento conforme desejado. A seção A.5.3 detalha sobre os modelos em língua portuguesa.

A seguir, será dado prosseguimento nos testes para avaliar a possibilidade de controle do robô, utilizando as funções do *SharpNLP* e o C#.

Testes com comandos do robô Curumim:

Para a segunda parte dos testes, uma vez que se trata de um robô móvel, foram escolhidos dois comandos do Curumim relacionados à sua capacidade de atuação: comando ***Move*** (mover) e comando ***Rotate*** (rotacionar). Porém, o robô não será efetivamente acionado por enquanto. Na documentação fornecida pelo fabricante, essas funções disponíveis na linguagem de programação C++ necessitam dos seguintes parâmetros:

- ***Move***(Distância X, Distância Y, Velocidade); e
- ***Rotate***(Ângulo, Velocidade).

Nota: posteriormente, durante o desenvolvimento, observou-se que para o acionamento real do robô Curumim na linguagem de programação C#, as funções disponíveis são diferentes e também requerem um conjunto ligeiramente diferente de parâmetros. As funções e parâmetros disponíveis no C# são: **andarReto**(Distância, Velocidade) e **rotacionar**(Ângulo, Velocidade). Para localizar as funções disponíveis nesta linguagem de programação e quais os parâmetros que requerem, foi feita análise diretamente na parte do código fonte atenciosamente fornecido pela sua equipe técnica, sem o qual seria virtualmente impossível acionar o Curumim pelo *software* em C#.

Dependendo da forma como o usuário está habituado a se expressar, eles podem ser escritos de variadas maneiras em linguagem natural. Alguns exemplos de frases curtas e elementares que podem ser escritas em inglês para expressar as ideias representadas pelos comandos ***Move*** e ***Rotate*** são:

- **Move** – *Walk 100 meters forward; Walk forward 100 meters; Move 100 m fast; Go fast 100 cm; Go back 100 mm; Go 200 meters quickly forward; etc*; e
- **Rotate** – *Return; Turn back; Turn; Turn left; Turn right; etc*.

Para realizar os testes direcionados aos comandos do robô Curumim, uma aplicação em linguagem C# foi elaborada. Para identificação dos comandos, foram considerados estes exemplos de comandos em linguagem natural usando o comando IF-THEN-ELSE, ou seja, um método determinístico simples baseado somente em lógica e escrita manualmente no código do programa. A solução de identificar os comandos escrevendo as palavras em Português diretamente no código do programa é insuficiente para o sistema robótico desejado, mas atende ao propósito de testar a vinculação de palavras do texto em linguagem natural aos comandos que controlam o robô.

A aplicação recém desenvolvida possui uma única interface. Com ela, os textos em linguagem curta e elementar podem ser inseridos no campo "Texto leigo", o processamento com as funções escolhidas do *SharpNLP* é executado internamente, a lógica de decisão é aplicada na saída dessas funções e o resultado final é exibido em campos apropriados, para avaliação do usuário. Seu aspecto geral é mostra na figura A.5.8.

Figura A.5.8 – Tela inicial da aplicação de teste de comandos ao Curumim.

Iniciando pelo comando ***Move***, ao ser inserido o primeiro texto de teste "*Walk 100 meters forward*" (Andar 100 metros para frente), a distância é convertida em milímetros internamente no *software*, uma velocidade padrão de valor 100 foi assumida por não ter sido informada nenhuma no texto e o resultado é como mostrado pela figura A.5.9.

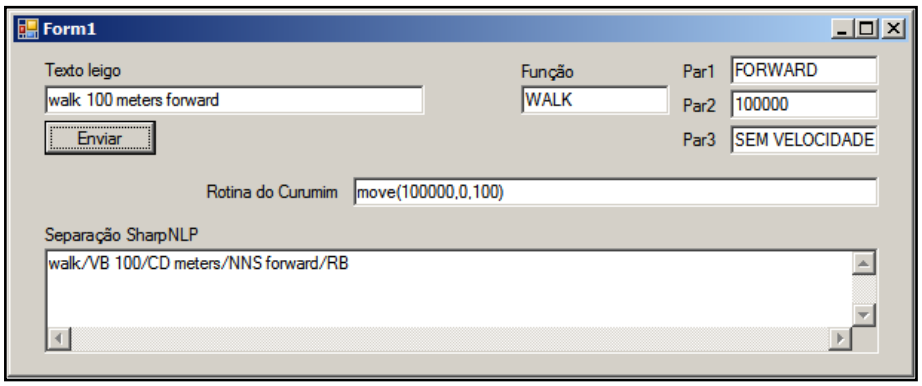


Figura A.5.9 – Primeiro teste para o comando *Move*.

Ao ser inserido o texto "*Walk forward*" (Andar para frete), uma distância padrão de valor 100 é assumida por não ter sido informada nenhuma no texto, o mesmo para a velocidade e o resultado é como mostra a figura A.5.10.

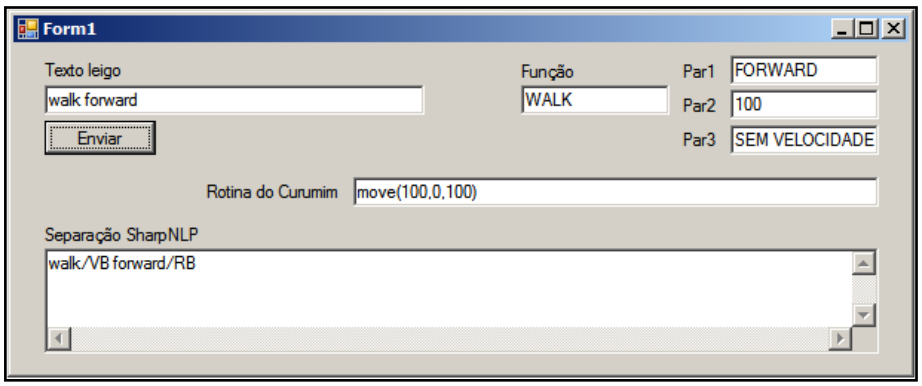


Figura A.5.10 – Segundo teste para o comando *Move*.

Testando com um simples texto "*Walk*" (Andar), valores padronizados são assumidos para todos os parâmetros, conforme se vê na figura A.5.11. Inclusive há uma incorreção no *software*, pois não há indicação de erro de distância não informada, mas tão somente a substituição pelo valor padronizado.

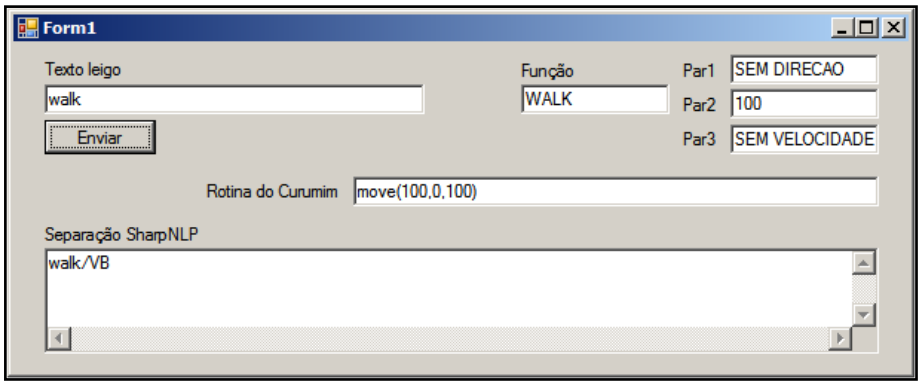


Figura A.5.11 – Terceiro teste para o comando *Move*.

O texto "*Go back 100 cm slowly*" (Ir para trás 100 cm vagarosamente) apresenta todos os parâmetros necessários do comando *Move* e nenhum aviso é gerado. Contudo, deve-se notar a indicação de inversão sentido, corretamente indicada pelo sinal negativo na composição do comando, na figura A.5.12.

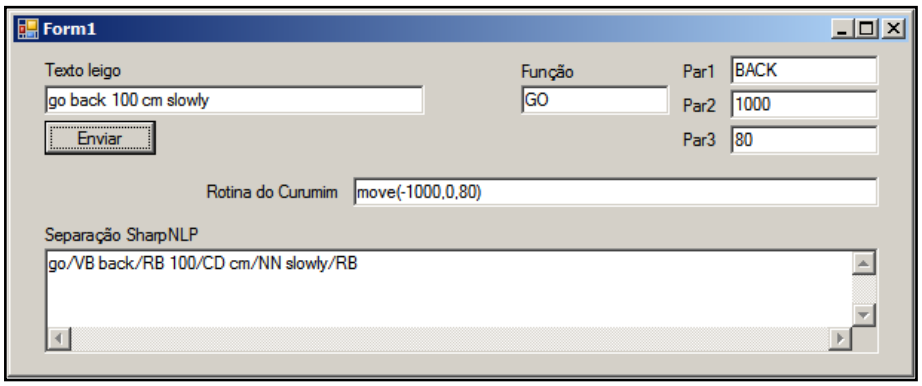


Figura A.5.12 – Quarto teste para o comando *Move*.

Noutro teste, "*Move 100 fast*" (Mover 100 rápido), apesar de ser estranho e incorreto do ponto de vista da linguagem natural, é suficiente para ser reconhecido pelo *software* como foi criado. Contém todos os parâmetros necessários, como mostra a figura A.5.13.

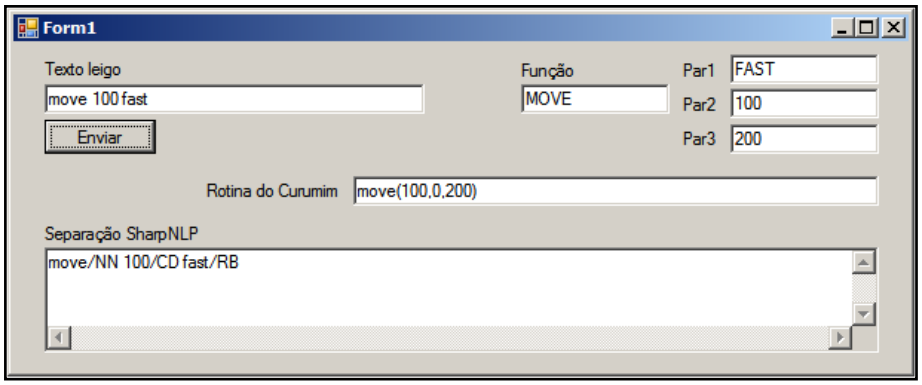


Figura A.5.13 – Quinto teste para o comando *Move*.

Num último teste para o comando *Move*, o texto "W*alk backward fast*" (Andar de ré rápido) não informa a distância e novamente o erro do *software* aparece ao simplesmente assumir um valor padrão sem mensagem de aviso. Mesmo assim, é reconhecida a inversão de sentido, o parâmetro velocidade foi informado e o resultado pode ser visto na figura A.5.14.

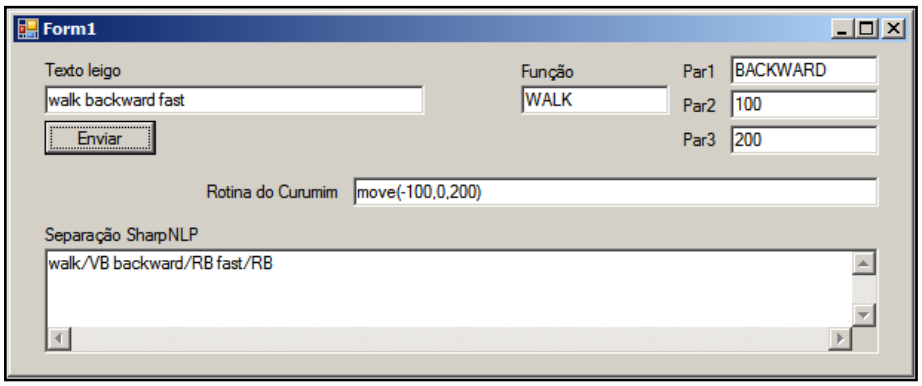


Figura A.5.14 – Sexto teste para o comando *Move*.

Partindo agora para o comando ***Rotate***, ao ser inserido o primeiro texto de teste "*Turn left*" (Virar à esquerda), o segundo teste "*Turn right*" (Virar á direita) e o terceiro teste "*Turn back*" (Virar para trás), os resultados são exibidos nas figuras A.5.15, A.5.16 e A.5.17 respectivamente. Estes três testes apresentam o único parâmetro necessário deste comando, já que se optou por fixar a velocidade de rotação para todos os casos.

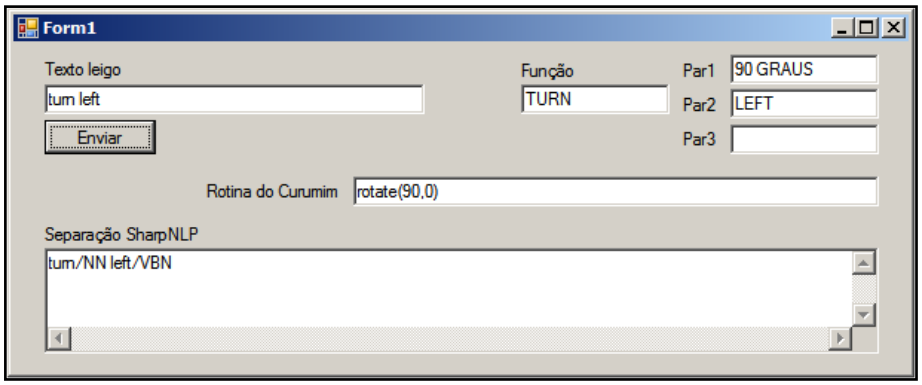


Figura A.5.15 – Primeiro teste para o comando *Rotate*.

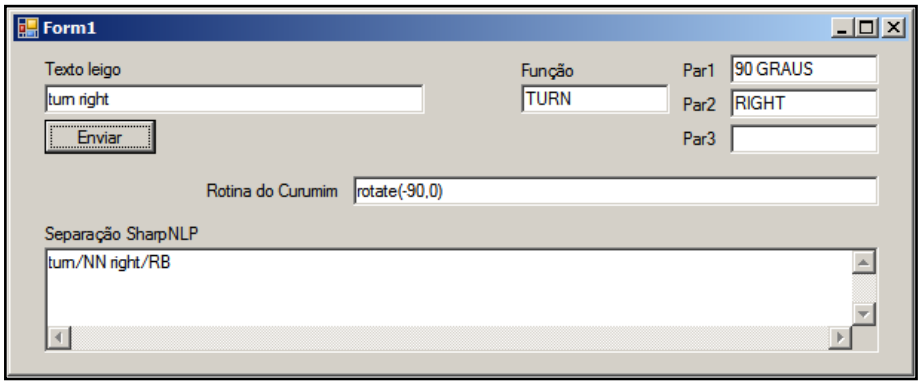


Figura A.5.16 – Segundo teste para o comando *Rotate*.

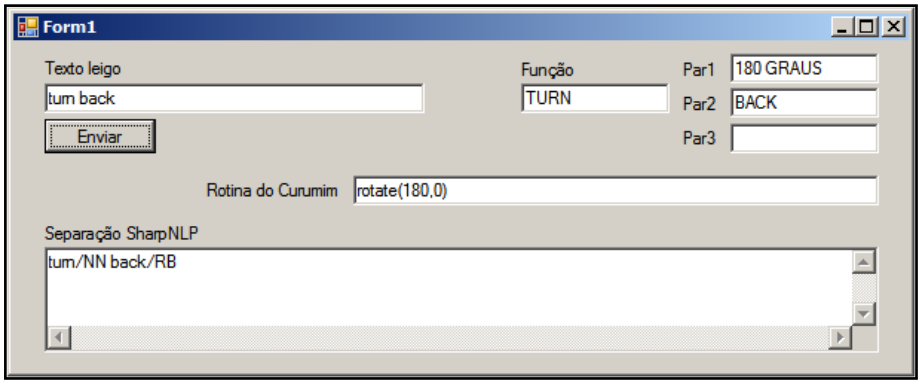


Figura A.5.17 – Terceiro teste para o comando *Rotate*.

Outros três testes, “*Return*” (Retornar), “*Back*” (Voltar) e “*Turn*” (Virar) são mostrados respectivamente nas figuras A.5.18, A.5.19 e A.5.20. No último deles, o texto não apresenta o parâmetro necessário e o *software* assume um valor padrão de 90 graus positivos.

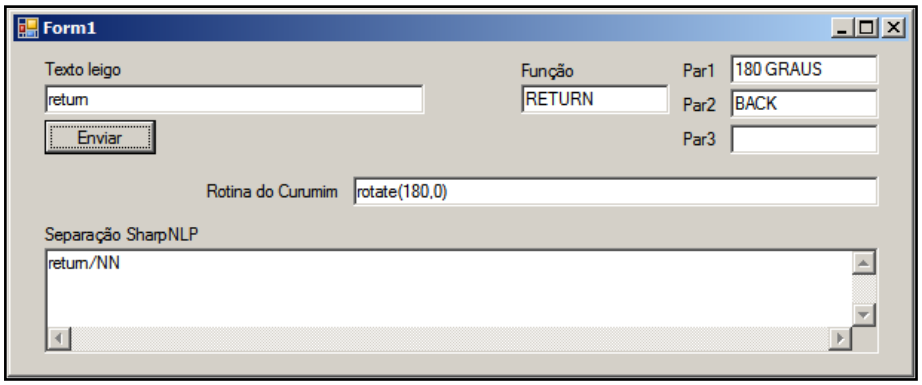


Figura A.5.18 – Quarto teste para o comando *Rotate*.

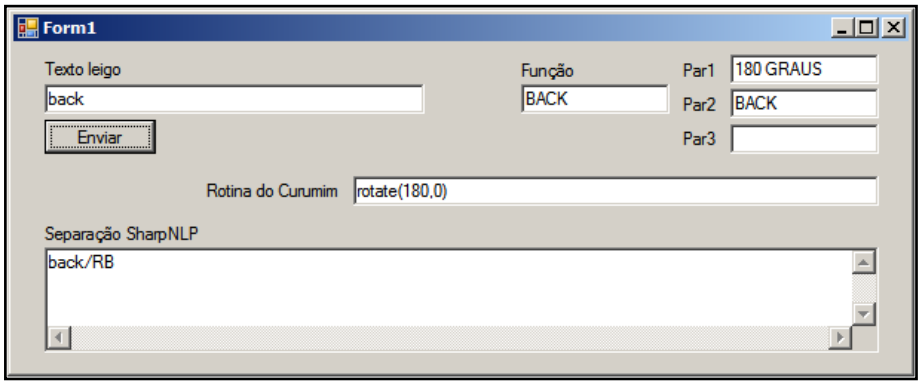


Figura A.5.19 – Quinto teste para o comando *Rotate*.

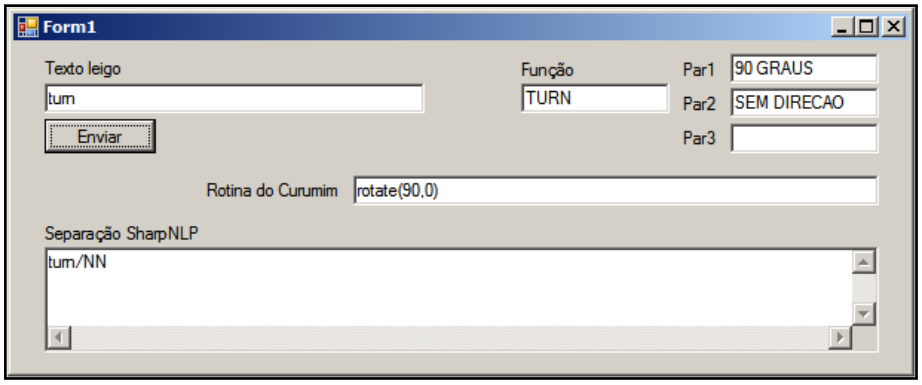


Figura A.5.20 – Sexto teste para o comando *Rotate*.

Análise e Conclusão finais:

As três funções escolhidas do *SharpNLP* são suficientes para atribuir a classificação gramatical mais provável para as palavras do texto informado pelo usuário. Em frases elaboradas com pouca complexidade linguística, estas funções parecem ser suficientes.

Da maneira como está funcionando, o *SharpNLP* somente pode ser usado com a língua Inglesa. Simplesmente utilizar palavras em Português não se observa nenhum resultado aproveitável. Deverá ser pesquisada forma de viabilizar seu uso na língua Portuguesa.

Durante os testes, se observou que há diferença em se escrever com maiúscula ou não a primeira letra da palavra inicial no texto em linguagem natural e a classificação gramatical resultante. Como forma de obter resultados mais consistentes, optou-se por forçar todo o texto livre que for inserido pelo usuário para letras minúsculas, função disponível na linguagem de programação C#, antes de submeter ao processamento pelas funções do *SharpNLP*.

Algumas palavras em Inglês, tanto quanto em Português, apresentam significado e classificação gramatical ambíguos, o que dificulta a identificação de comandos em linguagem natural. Não foi encontrado recurso no *SharpNLP* para trabalhar com sinônimos (das palavras).

Apesar dos recursos úteis do ponto de vista sintático do texto em linguagem natural, não foram encontradas funções para extração de significado no *SharpNLP*.

A forma escolhida para identificação de comandos, buscando por palavras específicas dentro do texto livre do usuário através de regras lógicas inseridas no código fonte do programa, mostrou-se funcional, porém pouco flexível para os propósitos desejados para o sistema robótico. Isto porque seria necessário modificar o programa sempre que uma nova palavra precisar ser incluída. Será preciso pesquisar forma mais flexível e genérica de processar o significado o texto em linguagem natural.

A.5.2 TESTE 2 – PRIMEIRA TENTATIVA DE CONTROLE DO ROBÔ

O objetivo é avaliar a separação do texto de linguagem natural composto de duas ou mais orações, processá-lo para identificar os comandos que trás e fazer a

comunicação com o robô Curumim, para que ele execute os comandos descritos nas orações.

No experimento anterior, foi determinada a maneira como identificar as classes gramaticais das palavras de uma oração e uma forma de identificar os comandos e parâmetros para movimentação do robô.

Neste experimento, o texto deverá ser separado em mais de uma oração e, em cada oração, identificar os comandos e parâmetros que serão enviados para o robô.

Os dois comandos abaixo serão usados como base para identificação dentro de um texto com duas ou mais orações, que serão colocados em fila para execução:

- ***Move***(Distância X, Distância Y, Velocidade); e
- ***Rotate***(Ângulo, Velocidade).

Como referências do tipo de texto em linguagem natural que podem ser utilizados, podem ser considerados:

- **Move** – *Walk 100 meters forward; Walk forward 100 meters; Move 100 m fast; Go fast 100 cm; Go back 100 mm; Go 200 meters quickly forward; etc*; e
- **Rotate** – *Return; Turn back; Turn; Turn left; Turn right; etc*.

Estes instruções livres podem usadas para compor um texto com várias orações para o robô executar. Em seguida, são listados alguns textos com duas ou mais orações que podem surgir durante a utilização:

- *Return. Go 200 meters quickly forward*.
- *Walk 100 meters forward. Turn back. Move 100 m fast. Turn Right*.
- *Turn Right. Go back 100 mm. Turn.*
- *Move 100 m fast. Turn back. Walk forward 100 meters. Turn. Move 100 meters forward. Back. Go fast 100 cm. Turn left*.

Execução dos testes

Para realizar os testes direcionados aos comandos do robô Curumim, a aplicação criada anteriormente em linguagem C# foi utilizada como base para uma nova. Para identificação dos comandos, foram considerados estes exemplos de texto livre usando a mesma solução baseada em método determinístico, escrito diretamente no código do programa, usando o comando IF-THEN-ELSE. Mesmo que insuficiente para o sistema robótico desejado, essa solução atende ao propósito dos testes.

Esta aplicação também possui uma única interface. Com ela, os textos do usuário podem ser inseridos, o processamento com as funções escolhidas do *SharpNLP* é executado internamente, a lógica de decisão é aplicada na saída dessas funções e o resultado final é listado em campos apropriados, para conhecimento do usuário.

Usando como referência as orações citadas acima, foram feitos alguns testes, sendo que a seguir serão mostradas cópias das telas com o texto e os comandos de saída que o robô deverá executar.

Usando o texto "*Return. Go 200 meters quickly forward*" (Retornar. Ir 200 metros rapidamente em frente.), o resultado é conforme a figura A.5.21.

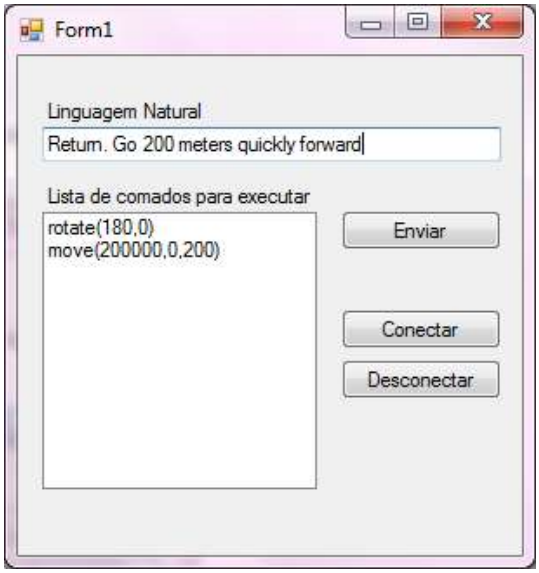


Figura A.5.21 – Primeiro teste de comandos do robô.

Usando o texto "*meters forward. Turn back. Move 100 m fast. Turn Right.*" (metros em frente. Virar para trás. Mover 100 m rápido. Virar à Direita.), sendo que uma incorreção foi propositadamente inserida, o resultado é conforme a figura A.5.22.

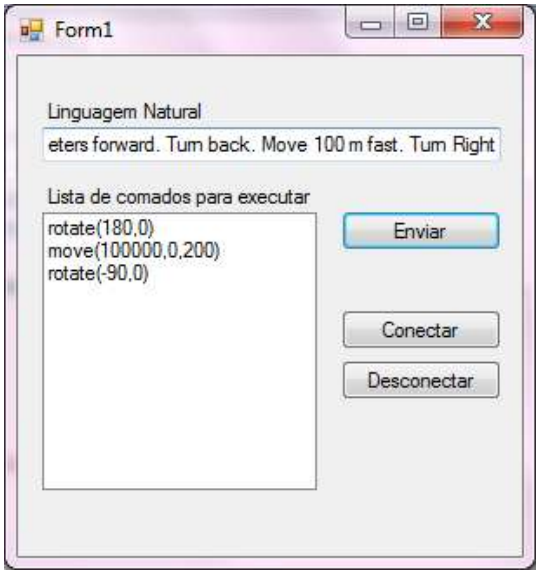


Figura A.5.22 – Segundo teste de comandos do robô.

O texto "*Turn Right. Go back 100 mm. Turn.*" (Virar à Direita. Voltar 100mm. Virar), é reconhecido conforme mostra a figura A.5.23.

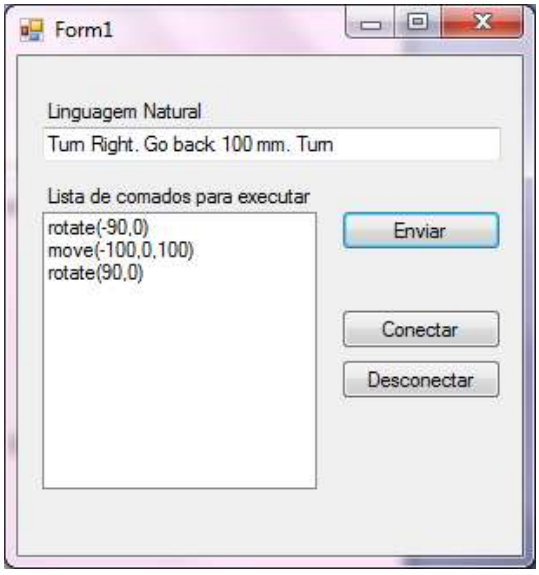


Figura A.5.23 – Terceiro teste de comandos do robô.

A longa instrução "*Move 100 meters quickly. Turn back. Go 100 meters forward. Turn right. Move 100 meters forward. Back. Go fast 100 cm. Turn left.*" (Mover 100 meters rapidamente. Virar para trás. Ir 100 metros em frente. Virar à direita. Mover

100 metros em frente. Voltar. Ir rápido 100 cm. Virar para esquerda.), é resulta na figura A.5.23.

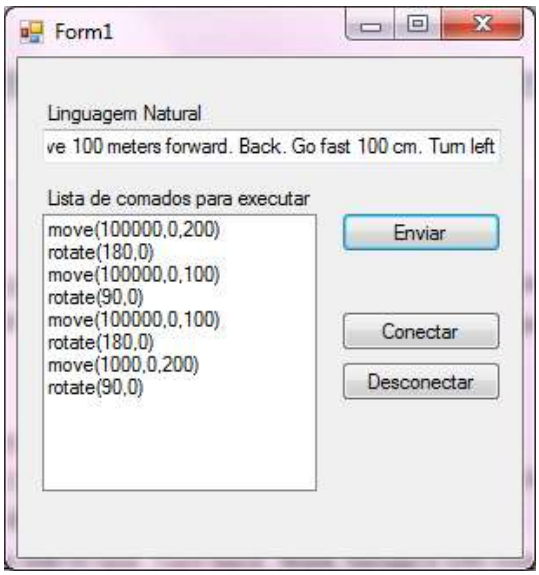


Figura A.5.24 – Quarto teste de comandos do robô.

Outras orações como "*Go back minus 20 cm*", "*Go fast and forward 1m*" e "*Walk back 30 cm*" também são interpretadas corretamente.

Análise e Conclusão:

Limitações do software:

Por definição de requisito inicial do *software*, são apenas identificadas orações separadas por '.' (ponto final). Se as instruções forem separadas por "," (vírgula) ou "*and*" (e), somente o primeiro comando é interpretado.

A interpretação do texto para reconhecimento dos comandos funciona desde que o verbo seja a primeira palavra da frase. Construções mais elaboradas como "*Please, walk 10 cm ahead*" (Por favor, ande 10 cm para frente.), "*I want you to go forward 20 cm*" (Quero que você vá para frente 20 cm), "*Can you move 1 m backwards for me, please?*" (Você poderia mover 1 m para trás para mim, por favor?) e "*Please, can you walk back 1 meter quickly?*" (Por favor, poderia andar para trás 1 metro rapidamente?) não são interpretadas.

Dificuldades encontradas:

Não houve dificuldade na separação do texto de linguagem natural em duas ou mais orações utilizando apenas regras lógicas definidas diretamente no programa em C# (método determinístico de processamento).

Na comunicação com o robô, uma dificuldade encontrada foi com relação a erros que o *software* retornava quando se pedia para abrir comunicação com uma porta COM (serial) do computador. Mesmo o painel de controle do sistema operacional *Windows* indicando a presença desta porta, a comunicação não era possível. Estudando melhor, foi identificado que o erro não se dava por problemas no código fonte, mas sim por prováveis motivos de compatibilidade de *hardware* com a máquina onde estava sendo desenvolvido o *software*. O mesmo *software* rodando em outro ambiente (máquina) funcionou corretamente.

Sugestões de melhorias:

- Incrementar o reconhecimento de palavras para comandos como "*quick*" (rápido), "*quickly*" (rapidamente), "*slow*" (lento), "*slowly*" (lentamente), "*backwards*" (para trás), "*ahead*" (à frente), "*run*" (correr), "*translate*" (transladar), "*dislocate*" (deslocar), "*displace*" (deslocar) e "*travel*" (viajar).

- Configurar velocidade padrão para 200, quando não informada. Configurar velocidade rápida como 300 e lenta como 100.

- O comando "*rotate*" (rotacionar) não reconhece mais que 90°. Ao receber comando para girar 180°, é preciso executar dois comandos de 90° na sequência. Aparentemente, trata-se de característica do *software* controlador (*driver*) do robô Curumim.

- Para melhorar a precisão da rotação, executar com velocidade mais baixa (50) e assim evitar problemas de escorregamento no piso e de inércia ao final da ação.

A.5.3 TESTE 3 – OS MODELOS EM PORTUGUÊS PARA O SHARPNLP.

Em pesquisa posterior para resolver o problema do *SharpNLP* estar funcionando em Inglês, e não em Português, foram encontrados modelos treinados

previamente em Português disponíveis para o *OpenNLP*. Por ser o *SharpNLP* uma portagem para C# do *Open*NLP (escrito em linguagem de programação Java), os modelos são compatíveis e podem ser convertidos utilizando ferramenta fornecida com o *SharpNLP*.

Feita a conversão, foram realizados alguns testes, com a aplicação de teste do *SharpNLP*, para saber como, e se, esses modelos poderiam ser aproveitados como originalmente disponibilizados.

Execução dos testes:

Diferentemente do Inglês, os modelos em Português apresentam um conjunto de "*Tags"* (marcações) diferentes. Após pesquisar a questão, encontrou-se que isso decorre do *Corpus*[7] utilizado para treinamento, que neste caso é o denominado Bosque. As *tags* mais comumente utilizadas são:

Dificuldades:

Comparativamente ao inglês, os modelos em português apresentaram resultados mais inconsistentes do que se esperava. As palavras aparecem demasiadamente classificadas como substantivo (/n), o que dificultará o processamento da linguagem. Aparentemente, tudo o que não é identificado pelo modelo é assumido como substantivo.

Exemplos de *Tags* do *SharpNLP*:

- Hhashhashsa: /n (substantivo)
- Metros: /n (substantivo)
- Frente: /n (substantivo)

Problemas Encontrados:

Um dos problemas encontrados foi inconsistência na classificação gramatical, caso seja inserida uma mesma palavra em maiúscula e minúscula. Outro problema foi com a acentuação, caso ela não seja feita corretamente, quando os modelos em Português parecem não identificar a palavra, classificando-a genericamente como substantivo.

[7] *Corpus*: conjunto grande de textos em linguagem natural, previamente preparado, normalmente por mãos humanas, para treinamento dos modelos estocásticos de processamento.

Exemplos:

- Rotacione: /prop (nome próprio - incorreto)
- rotacione: /n (substantivo - incorreto)
- rápido: /adv (advérbio - correto)
- rapido: /n (substantivo - incorreto)
- Rápido: /n (substantivo - incorreto)
- Rapido: /in (interjeição - incorreto)

Observar que nos quatro últimos exemplos a palavra “rápido” foi escrita de quatro modos diferentes.

Abaixo, as *tags* para as frases “olhe para cima” e “olhe para baixo”. Deve-se notar que as palavras “cima” e “baixo”, escritas de forma diferente, resultaram em classificações gramaticais diferentes no mesmo contexto de uso.

- Olhe**/v-fin** para**/prp** cima**/n**
- Olhe**/v-fin** para**/prp** Cima**/prop**
- Olhe**/v-fin** para**/prp** Baixo**/prop**
- Olhe**/v-fin** para**/prp** baixo**/adv**

Onde as *tags* são indicadas após cada palavra e separadas por “/” (barra) e significam:

- **v-fin**: verbo finito (conjugado no tempo e na pessoa);
- **prp**: preposição
- **n**: substantivo
- **prop**: nome próprio
- **adv**: advérbio

Análise e Conclusão:

Como forma de melhorar a consistência dos resultados obtidos, será mantida a solução de converter todo o texto livre do usuário em minúscula antes de processá-lo pelas funções do *SharpNLP*.

O resultado do processamento da linguagem natural pelo *SharpNLP*, assim como o *OpenNLP*, é estocástico. Isto significa que naturalmente apresentará certa taxa de erros, prevista teoricamente. Deverá ser pesquisado o que pode ser feito para melhorar os resultados, inclusive tendo-se em vista a aplicação desejada (o sistema robótico).